File Hijacking Vulnerability: The Elephant in the Room

Chendong Yu[1,2], Yang Xiao[1,2], Jie Lu[3], Yuekang Li[4], Yeting Li[1,2], Lian Li[3], Yifan Dong[1,2], Jian Wang[1,2], Jingyi Shi[1,2], Defang Bo[1,2], Wei Huo[1,2]

1. School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
2. Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
3. Institute of Computing Technology of the Chinese Academy of Sciences
4. University of New South Wales, Sydney, Australia

Security Boundary

Security Boundary	Security Goal
Network boundary	An unauthorized network endpoint cannot access or tamper with the code and data on a customer's device.
Kernel boundary	A non-administrative user mode process cannot access or tamper with kernel code and data. Administrator-to-kernel is not a security boundary.
Process boundary	An unauthorized user mode process cannot access or tamper with the code and data of another process.
AppContainer sandbox boundary	An AppContainer-based sandbox process cannot access or tamper with code and data outside of the sandbox based on the container capabilities
User boundary	A user cannot access or tamper with the code and data of another user without being authorized.
Session boundary	A user logon session cannot access or tamper with another user logon session without being authorized.
Web browser boundary	An unauthorized website cannot violate the same-origin policy, nor can it access or tamper with the native code and data of the Microsoft Edge web browser sandbox.
Virtual machine boundary	An unauthorized Hyper-V guest virtual machine cannot access or tamper with the code and data of another guest virtual machine; this includes Hyper-V Isolated Containers.
Virtual Secure Mode boundary	Data and code within a VSM trustlet or enclave cannot be accessed or tampered with by code executing outside of the VSM trustlet or enclave.

Security Boundary

Security Boundary	Security Goal
Network boundary	An unauthorized network endpoint cannot access or tamper with the code and data on a customer's device.
Kernel boundary	A non-administrative user mode process cannot access or tamper with kernel code and data. Administrator-to-kernel is not a security boundary.
Process boundary	An unauthorized user mode process cannot access or tamper with the code and data of another process.
AppContainer sandbox boundary	An AppContainer-based sandbox process cannot access or tamper with code and data outside of the sandbox based on the container capabilities
User boundary	A user cannot access or tamper with the code and data of another user without being authorized.
Session boundary	A user logon session cannot access or tamper with another user logon session without being authorized.
Web browser boundary	An unauthorized website cannot violate the same-origin policy, nor can it access or tamper with the native code and data of the Microsoft Edge web browser sandbox.
Virtual machine boundary	An unauthorized Hyper-V guest virtual machine cannot access or tamper with the code and data of another guest virtual machine; this includes Hyper-V Isolated Containers.
Virtual Secure Mode boundary	Data and code within a VSM trustlet or enclave cannot be accessed or tampered with by code executing outside of the VSM trustlet or enclave.

How to break security boundary?

File Hijacking Vulnerability (FHVuln): A type of security flaw where an attacker can breach the security boundaries by manipulating files, including file paths and contents, and they can result in severe security issues such as arbitrary code execution, privilege escalation, and data loss

Example Case

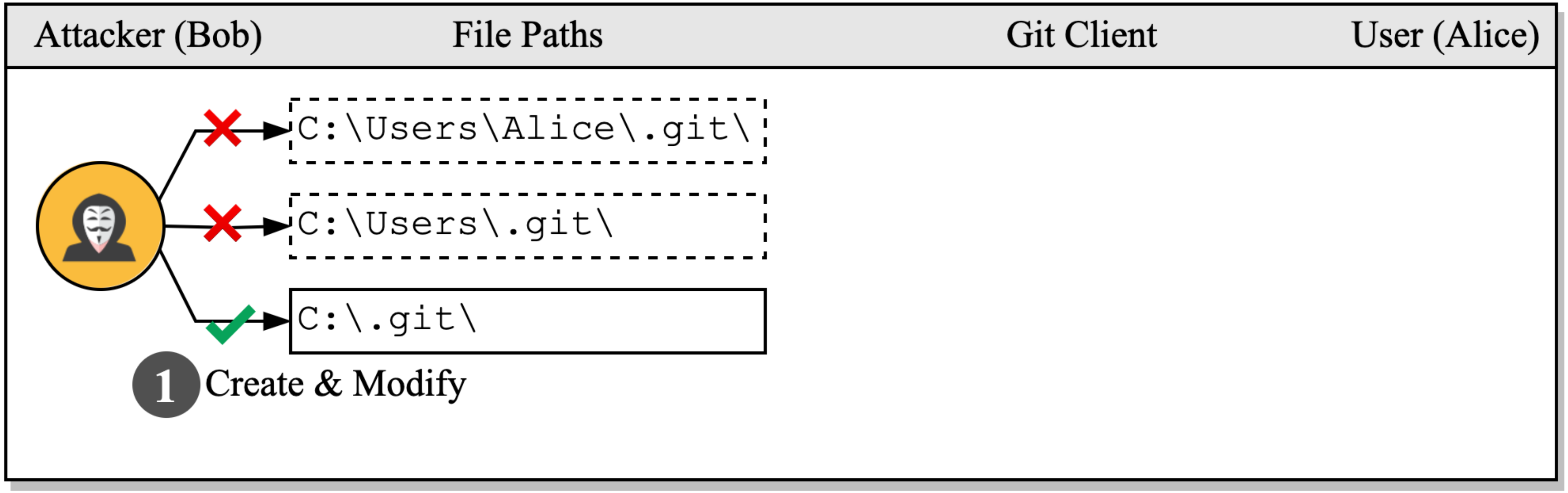


Fig: A FHVuln of Git identified by JERRY (CVE-2022-24765)

Example Case

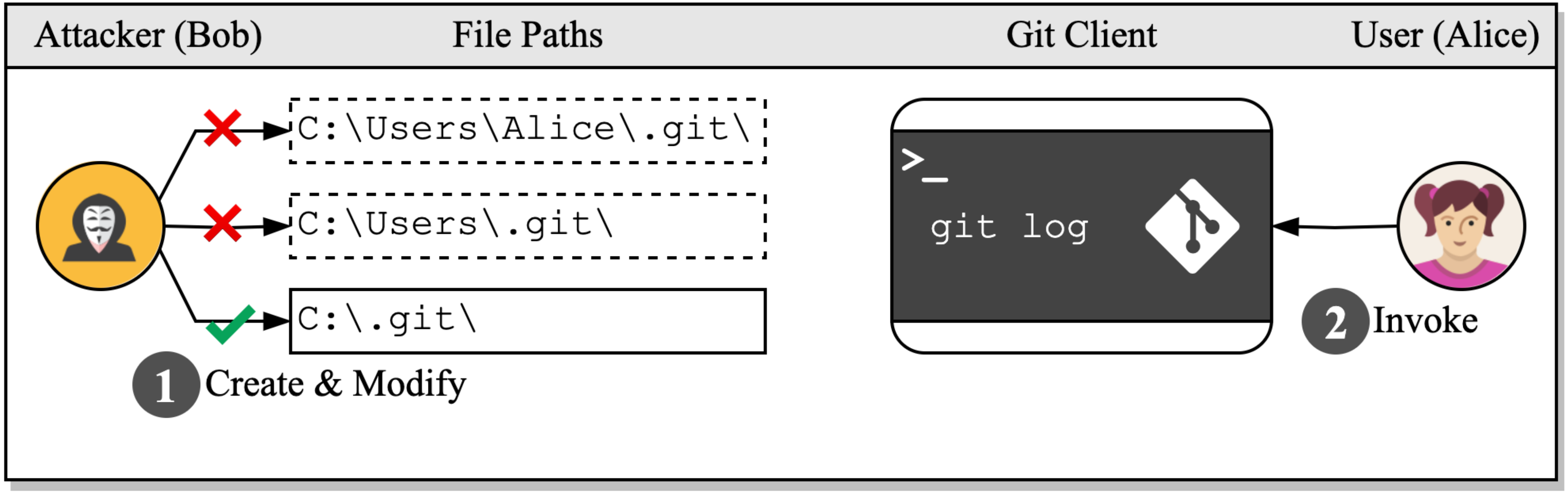


Fig: A FHVuln of Git identified by JERRY (CVE-2022-24765)

Example Case

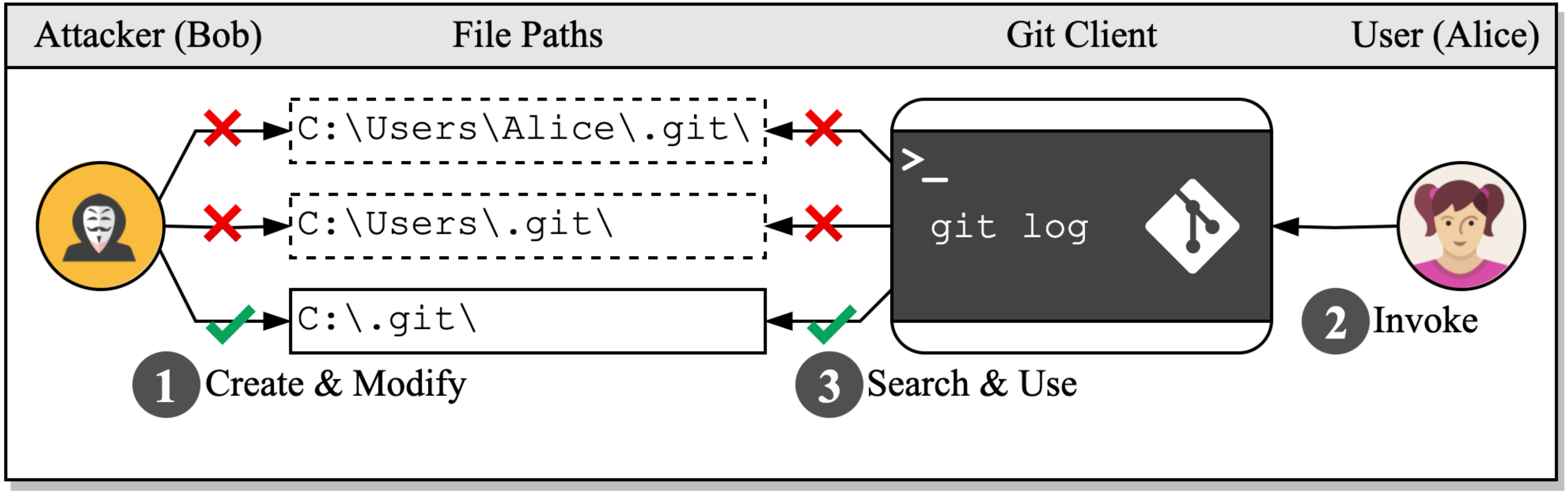


Fig: A FHVuln of Git identified by JERRY (CVE-2022-24765)

Example Case

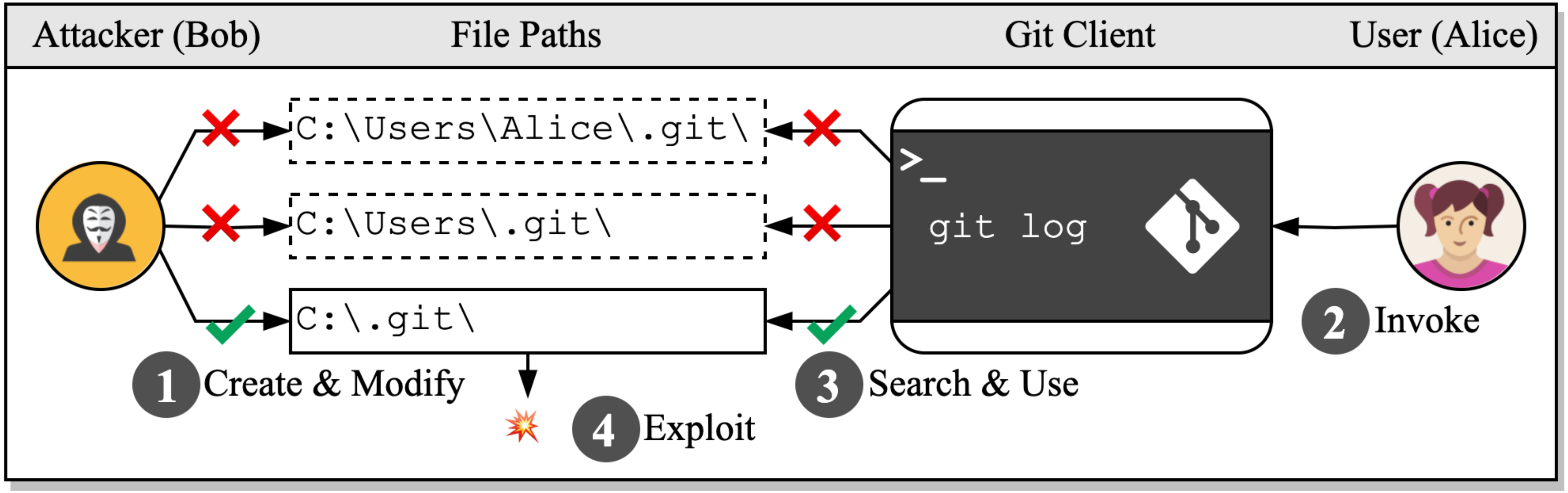
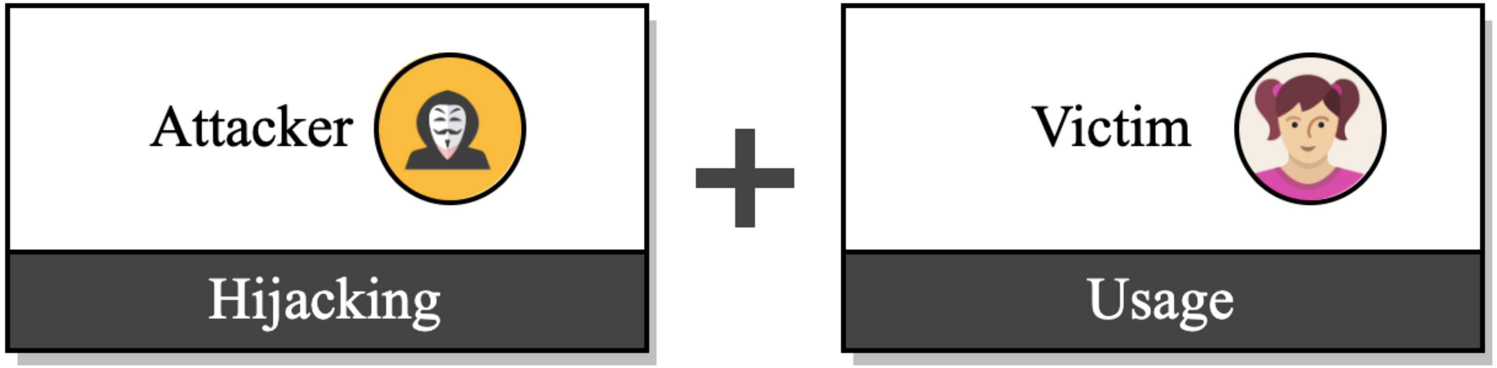


Fig: A FHVuln of Git identified by JERRY (CVE-2022-24765)

Threat Model & Study Questions



Attacker

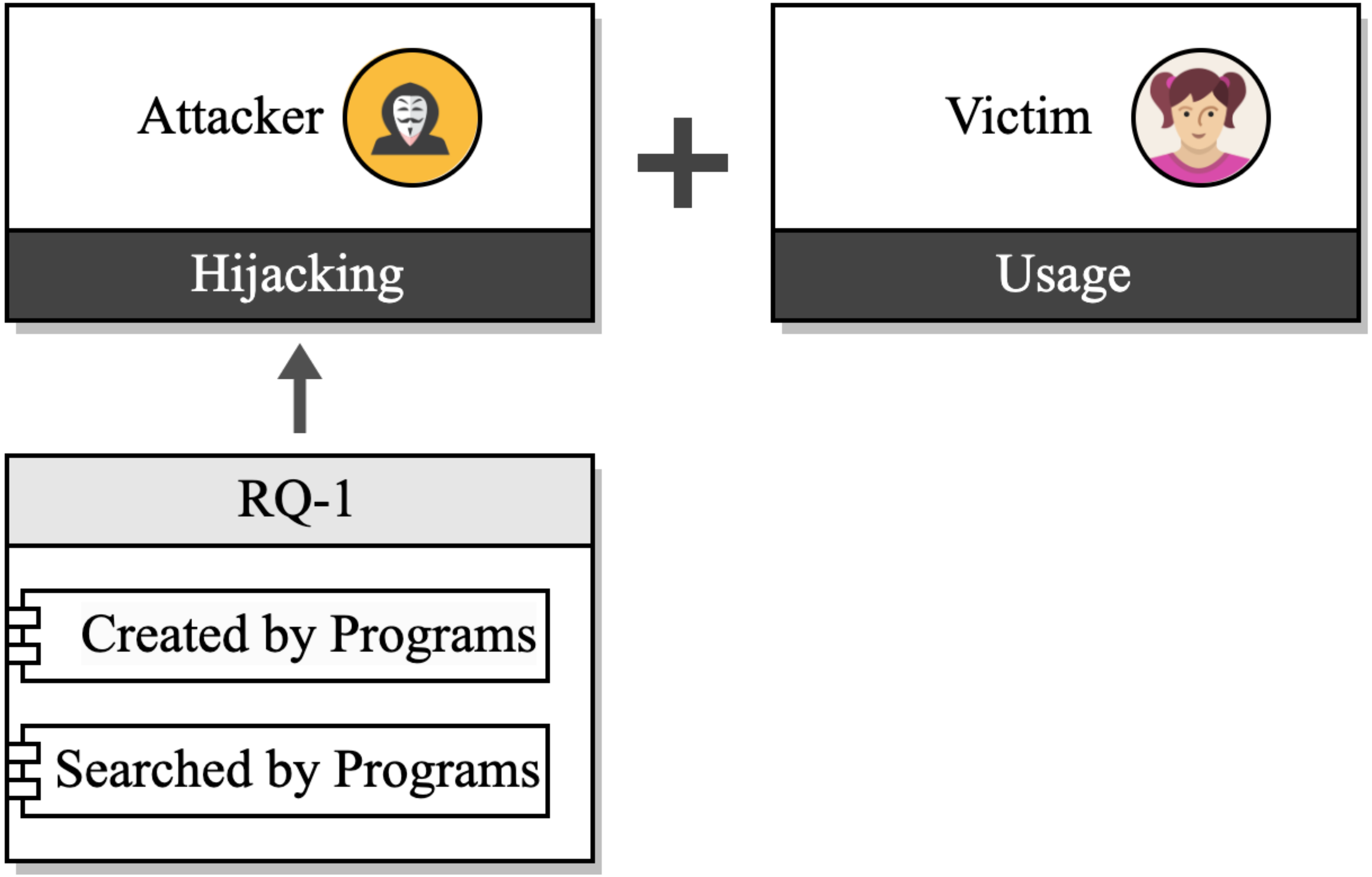
Hijacking

+

Victim

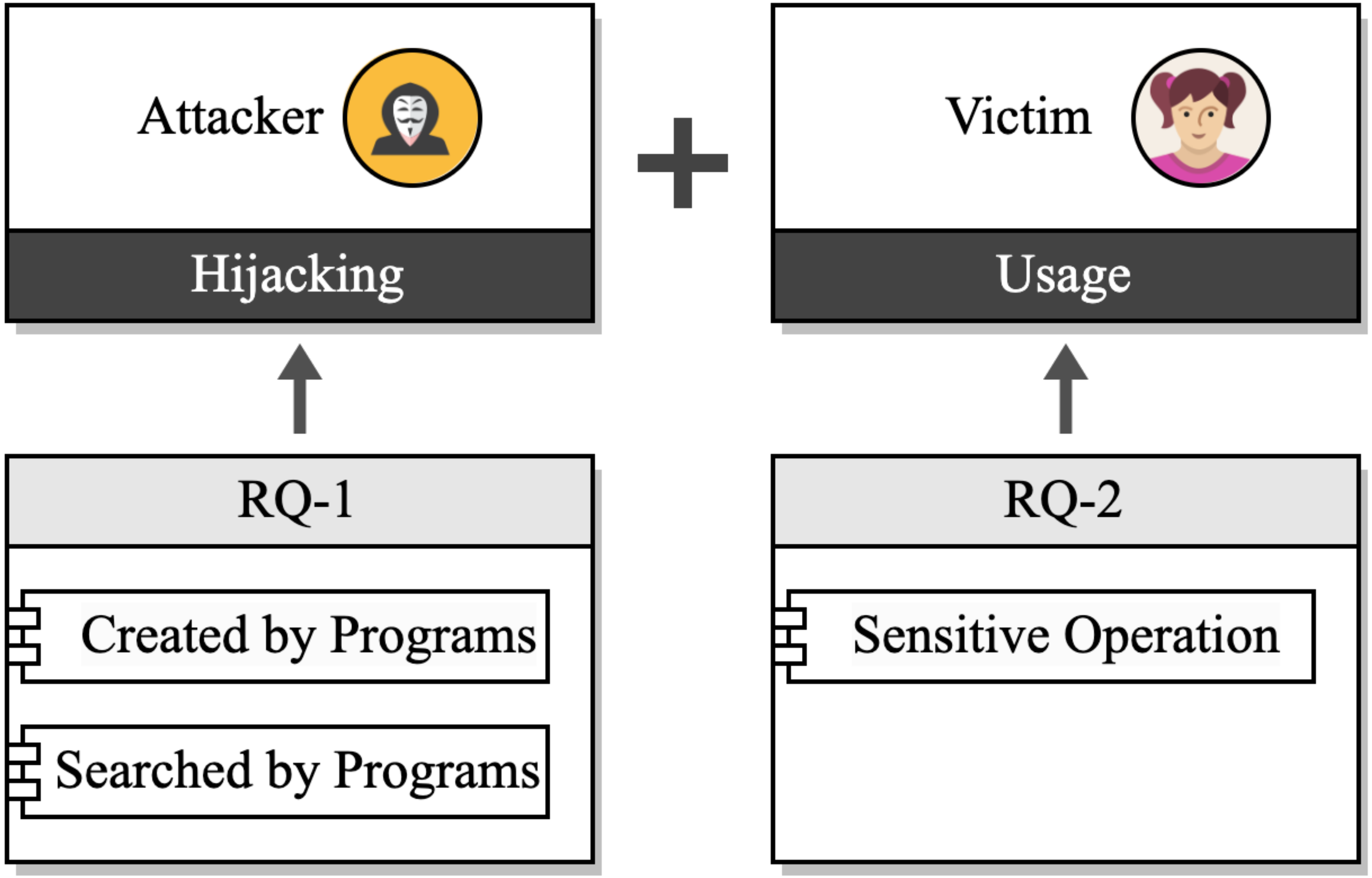
Usage

Threat Model & Study Questions



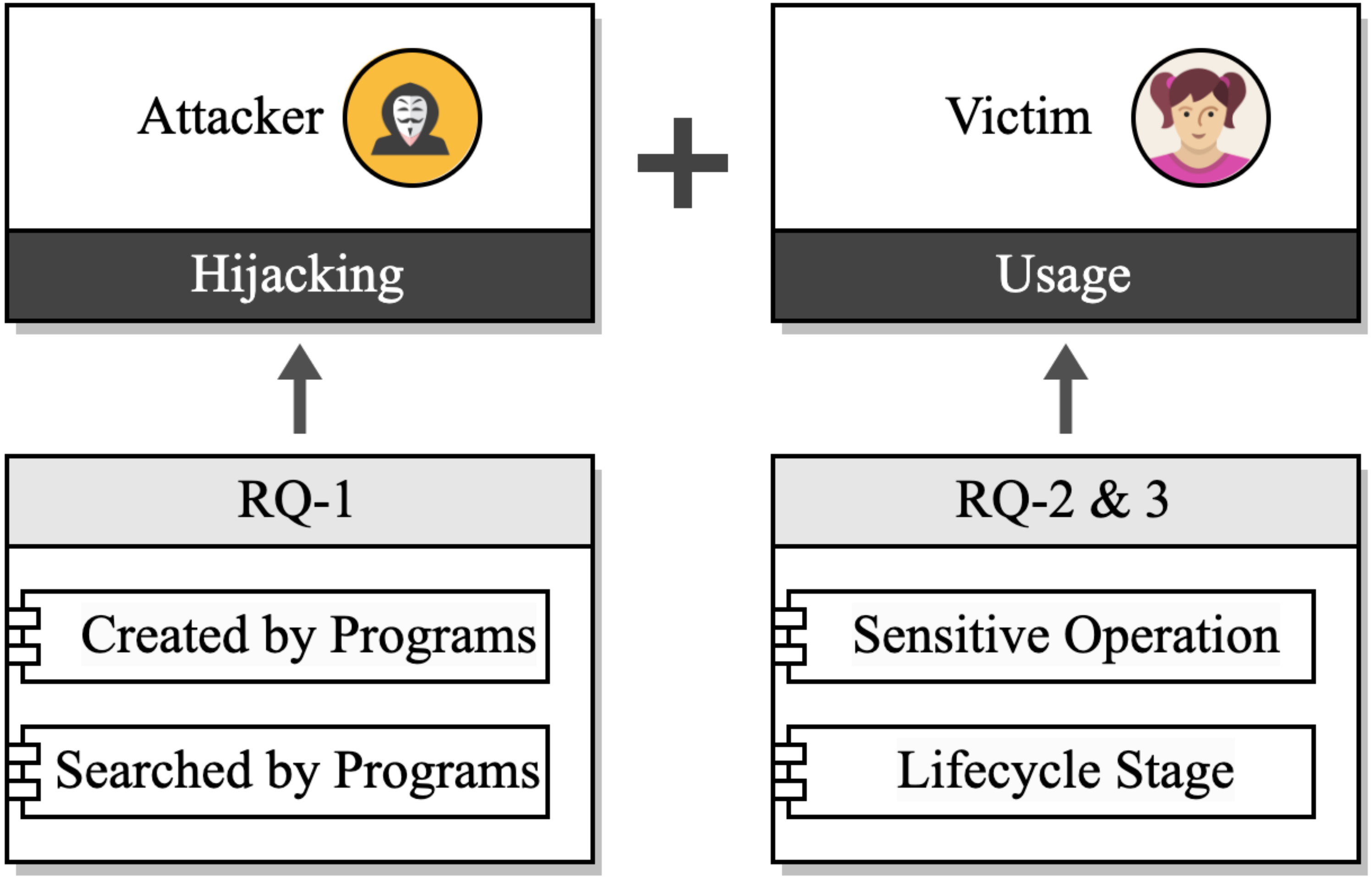
RQ1: What are the origins of the hijacked files?

Threat Model & Study Questions

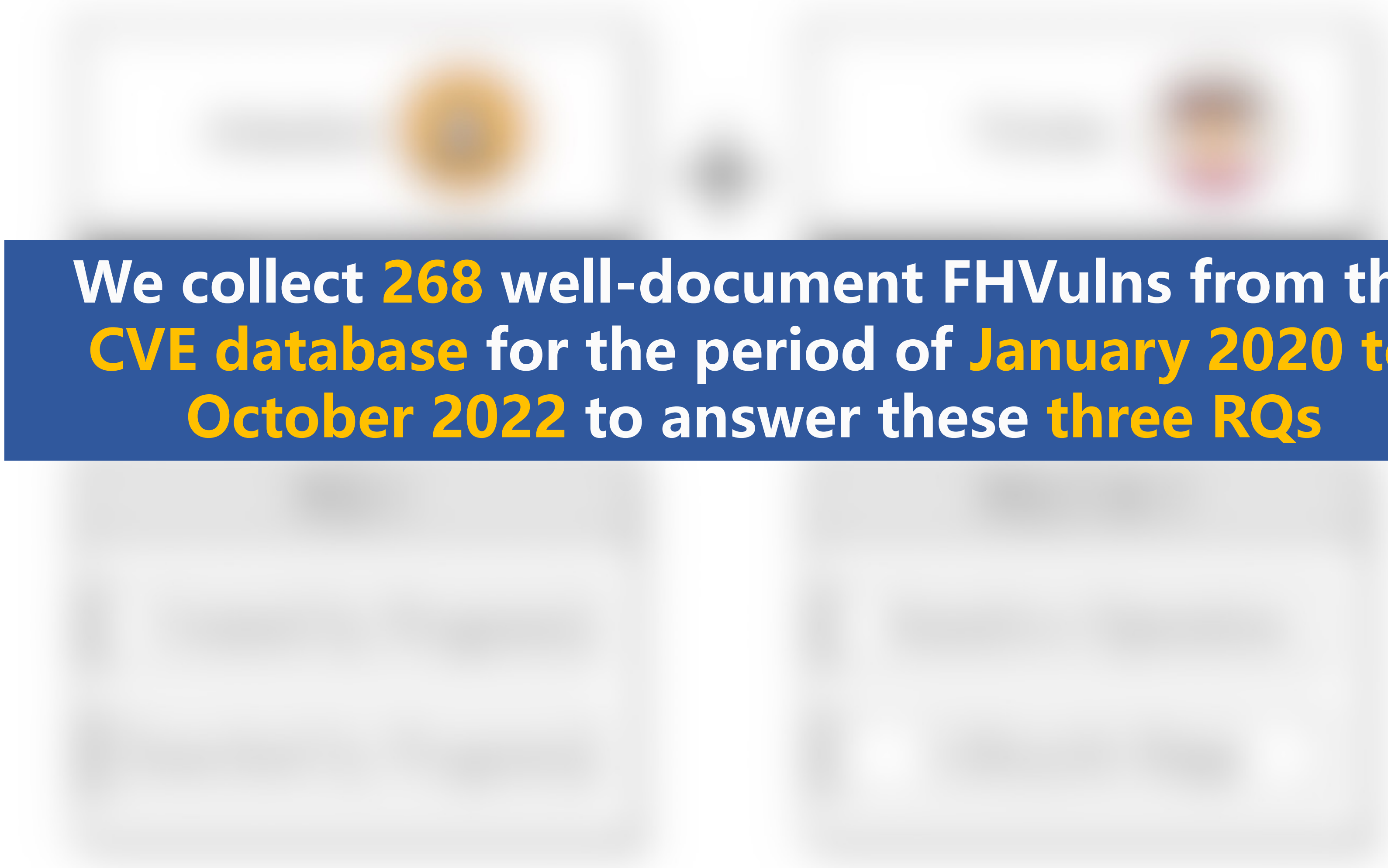


RQ2: What types of operations are dangerous vulnerability-triggering operations?

Threat Model & Study Questions



RQ3: When in the software lifecycle (installation, uninstallation, …) are FHVulns triggered?



We collect 268 well-document FHVulns from the CVE database for the period of January 2020 to October 2022 to answer these three RQs

RQ1 Origins of hijacked files

Observation 1: Most (89.9%) hijacked files are due to the **five search strategies** employed by the programs and the underlying operating systems, while the rest come from **files created by programs with weak permissions**.

RQ1 Origins of hijacked files

Observation 1: Most (89.9%) hijacked files are due to the **five search strategies** employed by the programs and the underlying operating systems, while the rest come from **files created by programs with weak permissions**.

	Strategy	Proportion
Created By Program	——	10.1%
Searched By Program	Path Search Order	3.4%
	Linux Path On Windows	4.5%
	Unquoted Path	17.1%
	Symbolic Links	19.4%
	Dynamically Loaded Libraries	44.5%

RQ2 Sensitive operations

Observation 2: There are **six types** of dangerous operations on hijacked files subject to file hijacking attacks. Among the six types of operations, **process creation** (28.4%) and **image loading** (45.1%) are most frequently exploited. The other four types of dangerous operations are **moving** (1.1%), **reading** (7.1%), **creating** (8.2%), and **deleting** (10.1%).

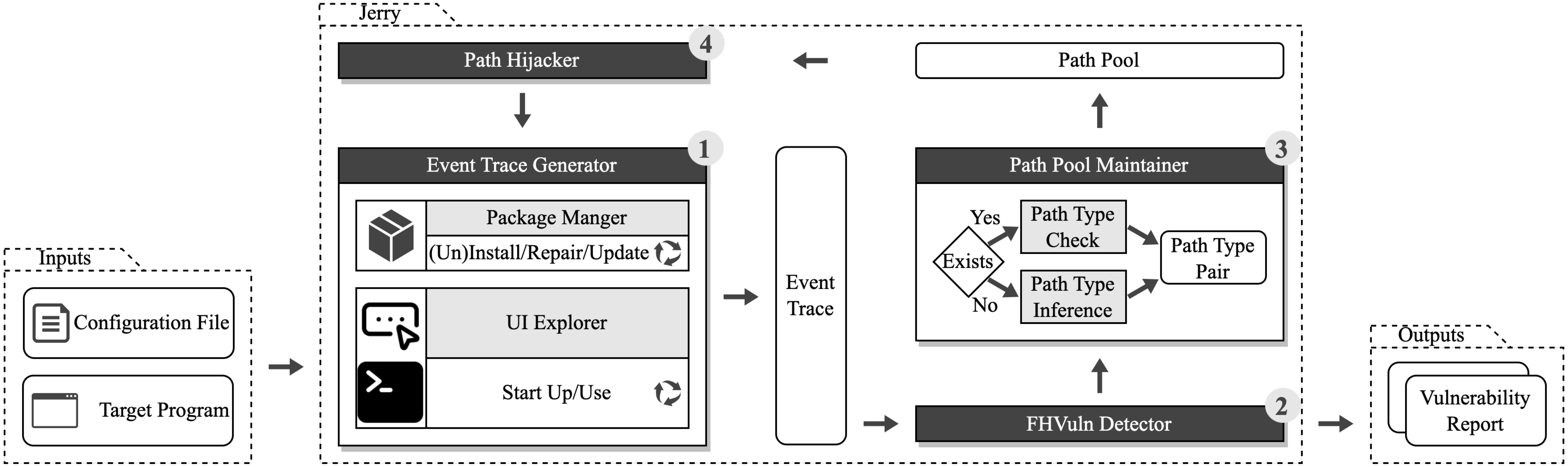
RQ2 Sensitive operations

Observation 2: There are **six types** of dangerous operations on hijacked files subject to file hijacking attacks. Among the six types of operations, **process creation** (28.4%) and **image loading** (45.1%) are most frequently exploited. The other four types of dangerous operations are **moving** (1.1%), **reading** (7.1%), **creating** (8.2%), and **deleting** (10.1%).

RQ3 Software lifecycle

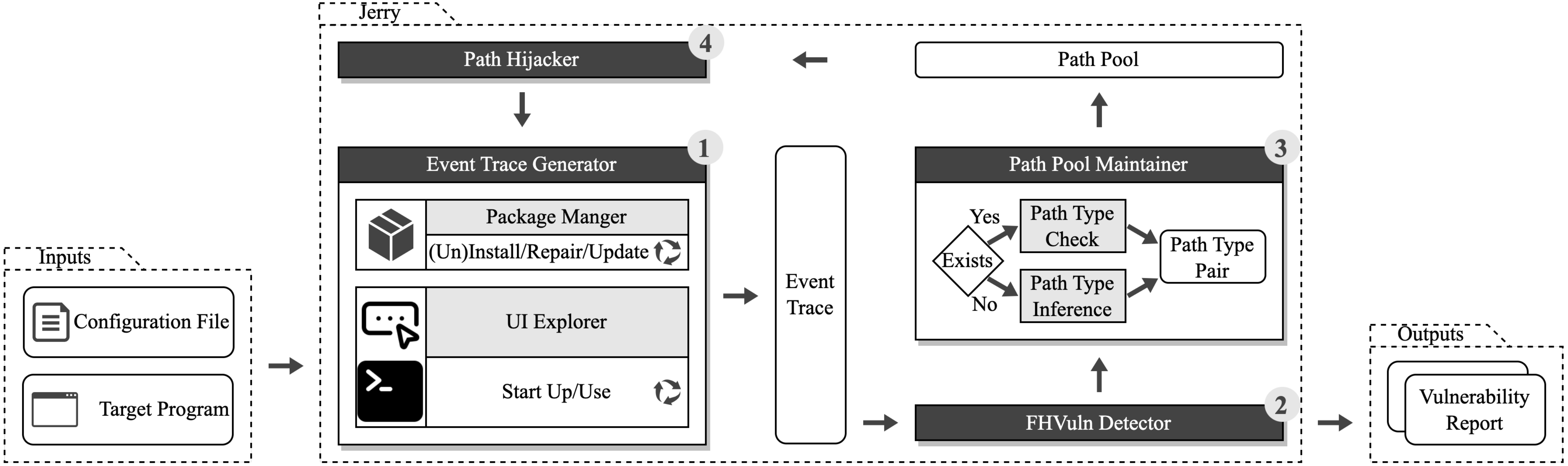
Observation 3: While the majority (62.3%) of FHVulns are exploited during the **Starting up** stage, FHVulns can be triggered at any stage during the software lifecycle, i.e., **Installation** (17.2%), **Uninstallation** (4.5%), **Updating** (1.9%), **Repairing** (3.7%) and **Usage** (10.4%).

Overview of JERRY



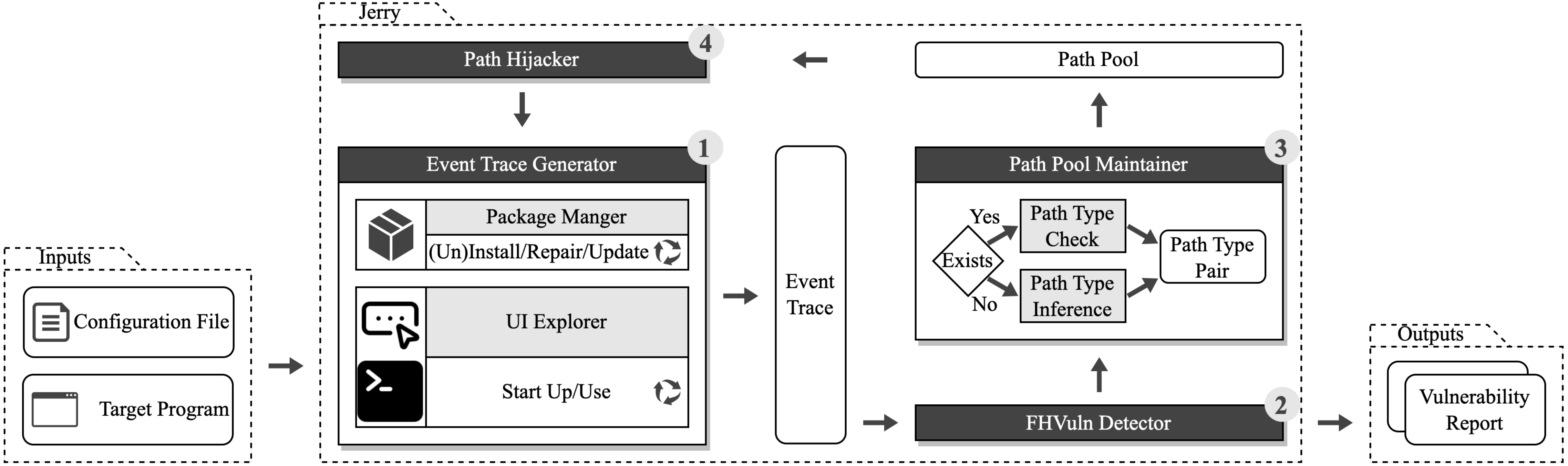
- **Event Trace Generator:** Execute the target program at each stage and records executed file operation traces

Overview of JERRY



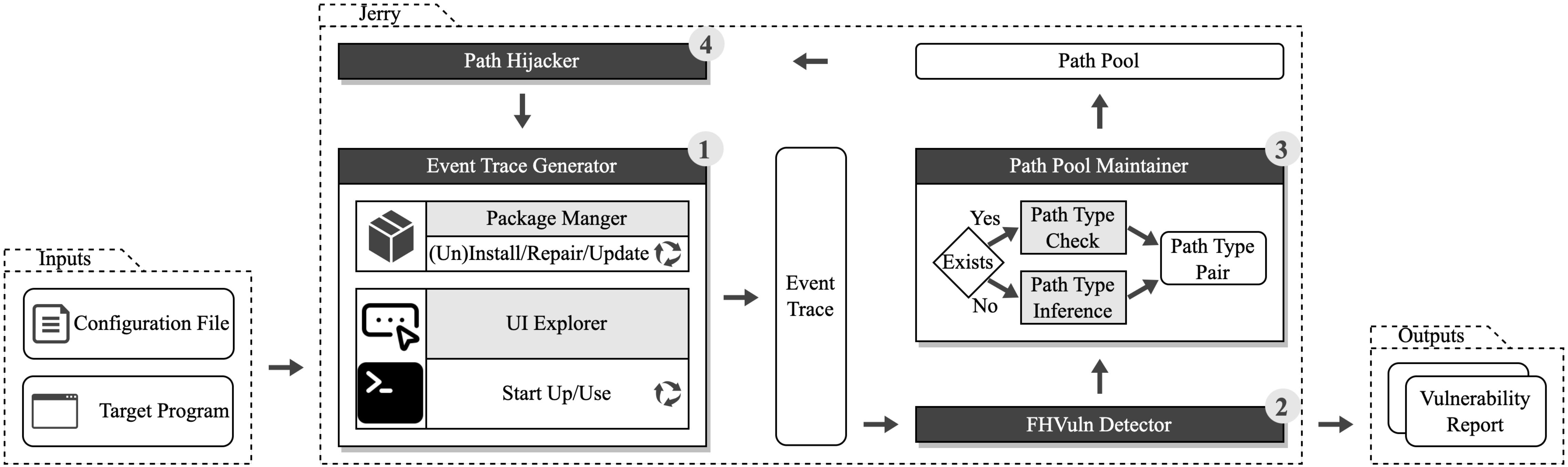
- **Event Trace Generator:** Execute the target program at each stage and records executed file operation traces
- **FHVuln Detector:** Examine each execution traces and a FHVuln will be reported if the trace performs dangerous operations on hijacked files.

Overview of JERRY



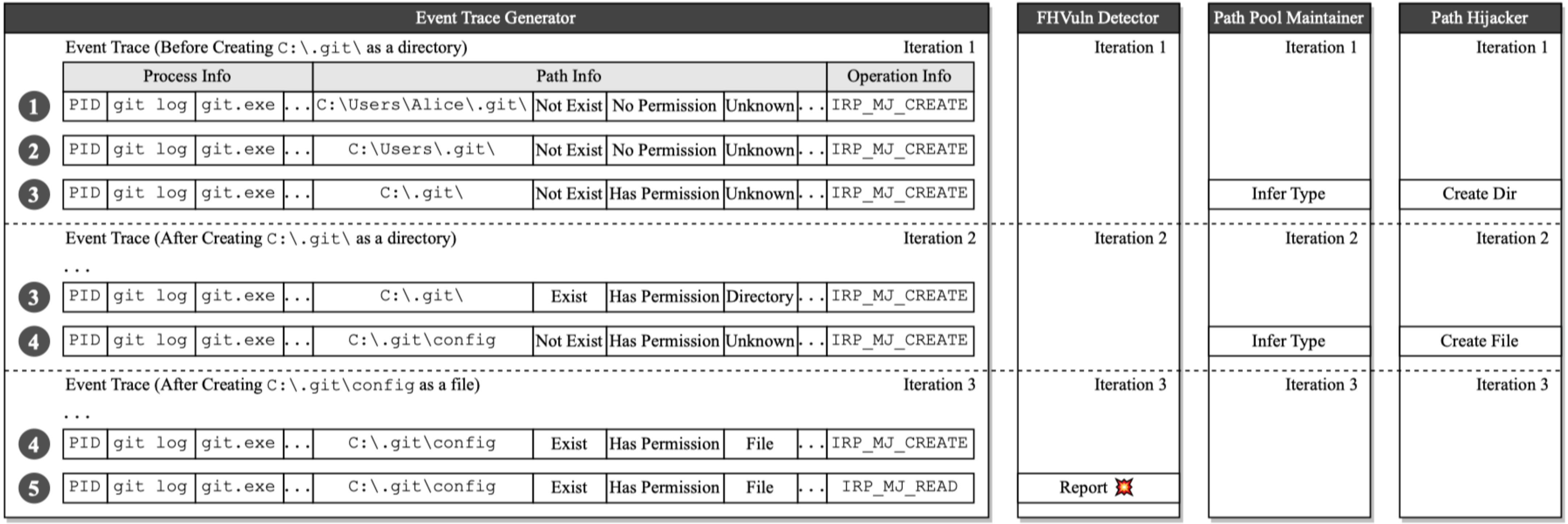
- **Event Trace Generator:** Execute the target program at each stage and records executed file operation traces
- **FHVuln Detector:** Examine each execution traces and a FHVuln will be reported if the trace performs dangerous operations on hijacked files.
- **Path Pool Maintainer:** Collect files encountered in the event trace and puts them into the path pool. Check if the file refers to a normal file or a directory.

Overview of JERRY



- **Event Trace Generator:** Execute the target program at each stage and records executed file operation traces
- **FHVuln Detector:** Examine each execution traces and a FHVuln will be reported if the trace performs dangerous operations on hijacked files.
- **Path Pool Maintainer:** Collect files encountered in the event trace and puts them into the path pool. Check if the file refers to a normal file or a directory.
- **Path Hijacker:** Hijack file or file path as an attacker

Event Trace Generator



Event Trace (Before Creating C:\.git\ as a directory) Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	. . .	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	. . .	IRP_MJ_CREATE
2	PID	git log	git.exe	. . .	C:\Users\.git\	Not Exist	No Permission	Unknown	. . .	IRP_MJ_CREATE
3	PID	git log	git.exe	. . .	C:\.git\	Not Exist	Has Permission	Unknown	. . .	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory) Iteration 2

. . .

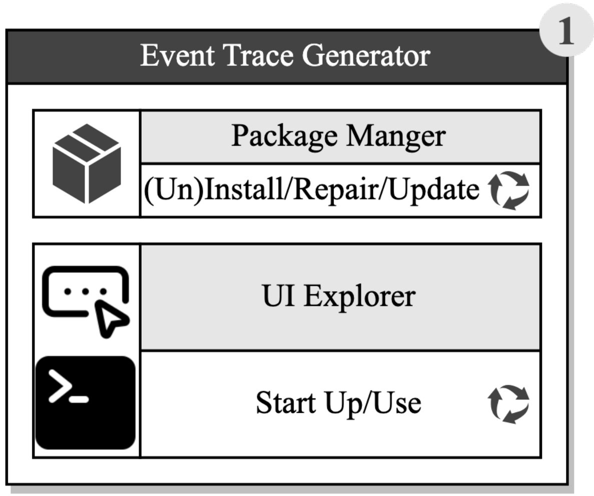
3	PID	git log	git.exe	. . .	C:\.git\	Exist	Has Permission	Directory	. . .	IRP_MJ_CREATE
4	PID	git log	git.exe	. . .	C:\.git\config	Not Exist	Has Permission	Unknown	. . .	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\config as a file) Iteration 3

. . .

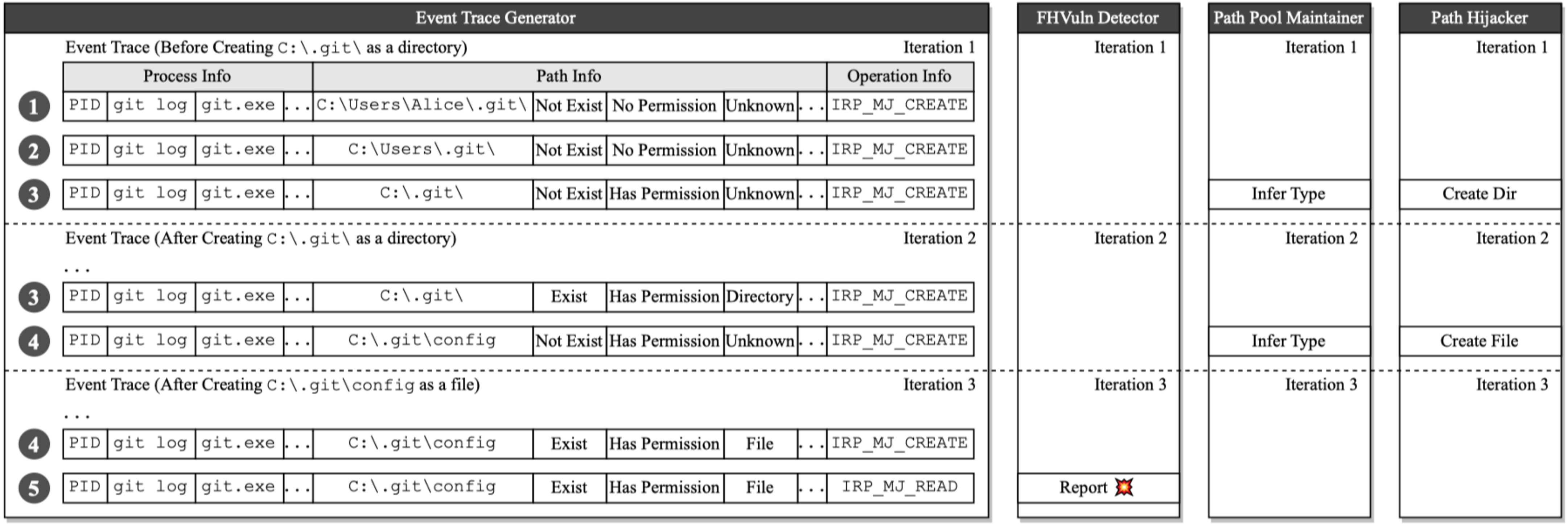
4	PID	git log	git.exe	. . .	C:\.git\config	Exist	Has Permission	File	. . .	IRP_MJ_CREATE
5	PID	git log	git.exe	. . .	C:\.git\config	Exist	Has Permission	File	. . .	IRP_MJ_READ

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking



- **Event Trace Generator:** Execute the target program at each stage and records executed file operation traces

Event Trace Generator



Event Trace Generator

Event Trace (Before Creating C:\.git\ as a directory) Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory) Iteration 2

...

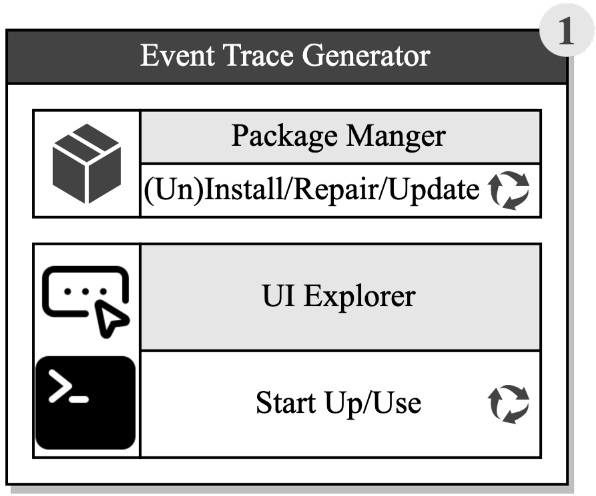
3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\config as a file) Iteration 3

...

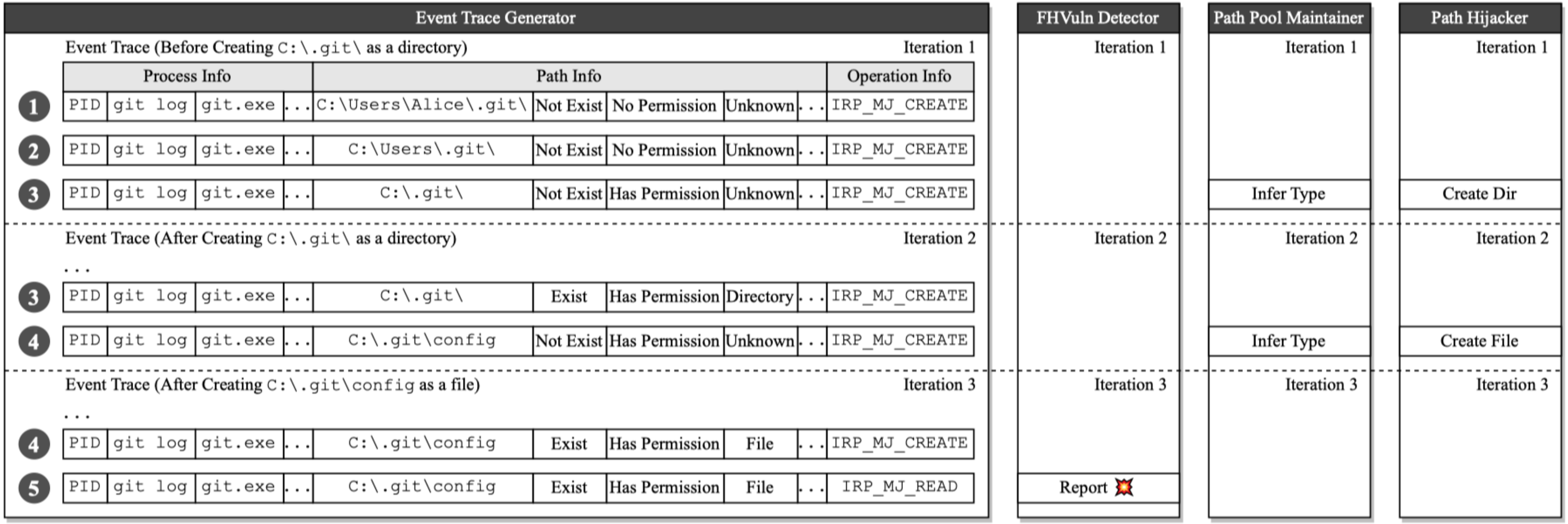
4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking



- **Event Trace Generator:** Execute the target program at each stage and records executed file operation traces
- **(Un)Install/Repair/Update:**
  - Automate execution with package manager

Event Trace Generator



Event Trace (Before Creating C:\.git\ as a directory) Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory) Iteration 2

...

3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

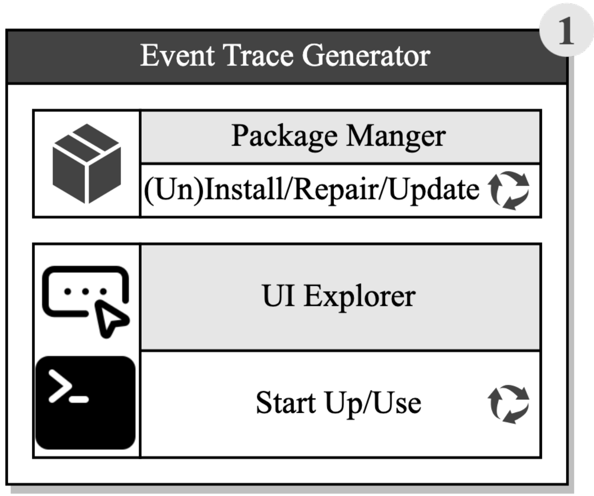
Event Trace (After Creating C:\.git\config as a file) Iteration 3

...

4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

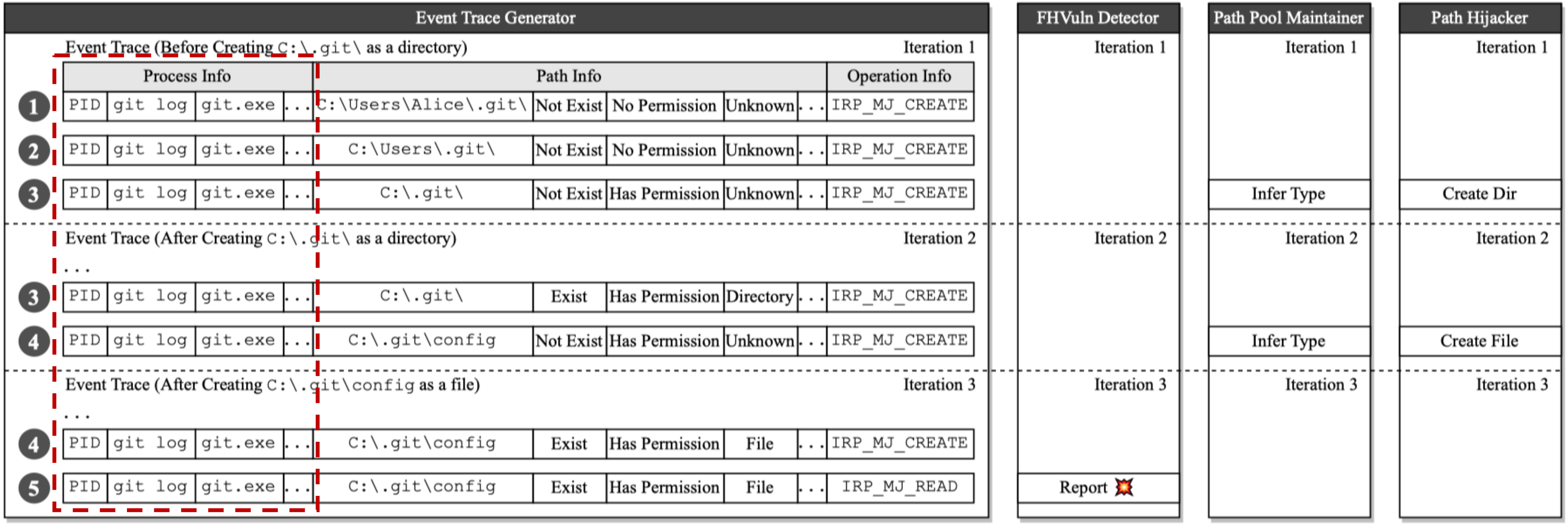
FHVuln Detector	Path Pool Maintainer	Path Hijacker
Iteration 1	Iteration 1	Iteration 1
	Infer Type	Create Dir
Iteration 2	Iteration 2	Iteration 2
	Infer Type	Create File
Iteration 3	Iteration 3	Iteration 3
Report 💥		

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking



- **Event Trace Generator:** Execute the target program at each stage and records executed file operation traces
- **(Un)Install/Repair/Update:**
  - Automate execution with package manager
- **Start Up/Use:**
  - For GUI, simple interactive like bottom click
  - For command line, read config from configure file (e.g. git log)

Event Trace Generator



Event Trace (Before Creating `C:\.git\` as a directory) Iteration 1

	Process Info				Path Info						Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...		IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...		IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...		IRP_MJ_CREATE

Event Trace (After Creating `C:\.git\` as a directory) Iteration 2

...

3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating `C:\.git\config` as a file) Iteration 3

...

4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

FHVuln Detector: Iteration 1; Iteration 2; Iteration 3; Report 💥

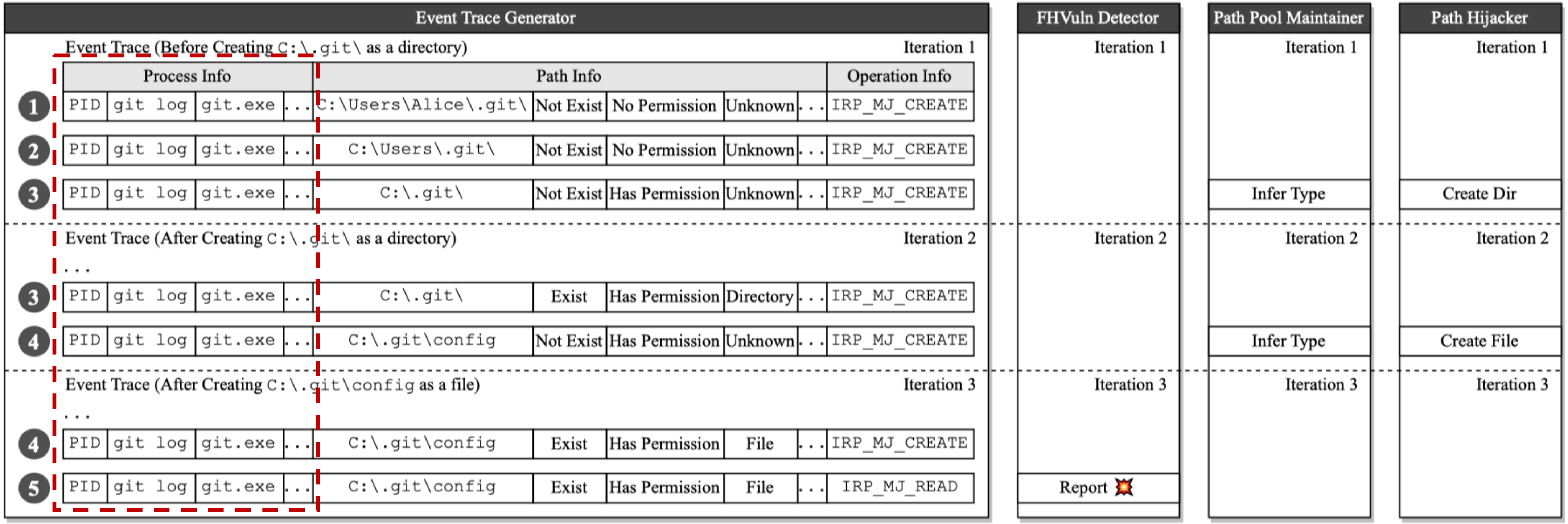
Path Pool Maintainer: Iteration 1; Infer Type; Iteration 2; Infer Type; Iteration 3

Path Hijacker: Iteration 1; Create Dir; Iteration 2; Create File; Iteration 3

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking

- **Event Trace Generator:** Execute the target program at each stage and records executed file operation traces

Event Trace Generator



Event Trace Generator

Event Trace (Before Creating C:\.git\ as a directory) Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory) Iteration 2

...

3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\config as a file) Iteration 3

...

4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

FHVuln Detector	Path Pool Maintainer	Path Hijacker
Iteration 1	Iteration 1	Iteration 1
	Infer Type	Create Dir
Iteration 2	Iteration 2	Iteration 2
	Infer Type	Create File
Iteration 3	Iteration 3	Iteration 3
Report 💥		

(a) Event Trace Generation

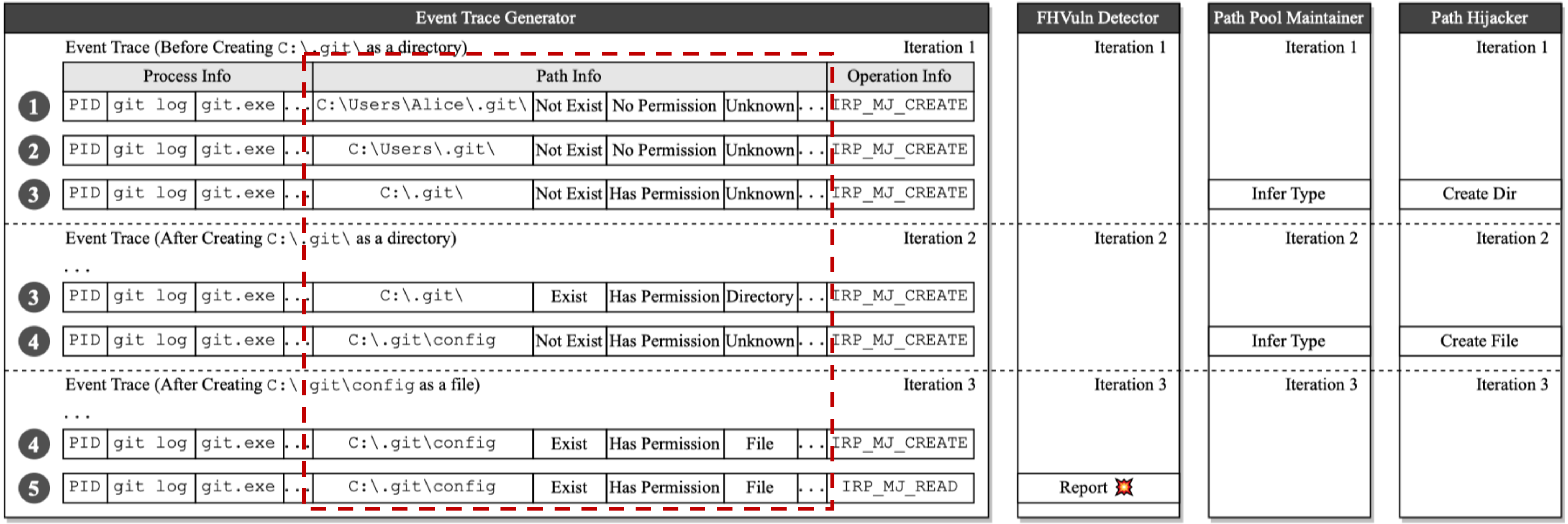
(b) FHVuln Detection

(c) Path Analysis

(d) Path Hijacking

- **Event Trace Generator:** Execute the target program at each stage and records executed file operation traces
  - **Monitor Process Info:** process id、command line option (GUI events)、executed program

Event Trace Generator



Event Trace Generator

Event Trace (Before Creating C:\.git\ as a directory) — Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	. . .	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	. . .	IRP_MJ_CREATE
2	PID	git log	git.exe	. . .	C:\Users\.git\	Not Exist	No Permission	Unknown	. . .	IRP_MJ_CREATE
3	PID	git log	git.exe	. . .	C:\.git\	Not Exist	Has Permission	Unknown	. . .	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory) — Iteration 2

. . .

3	PID	git log	git.exe	. . .	C:\.git\	Exist	Has Permission	Directory	. . .	IRP_MJ_CREATE
4	PID	git log	git.exe	. . .	C:\.git\config	Not Exist	Has Permission	Unknown	. . .	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\config as a file) — Iteration 3

. . .

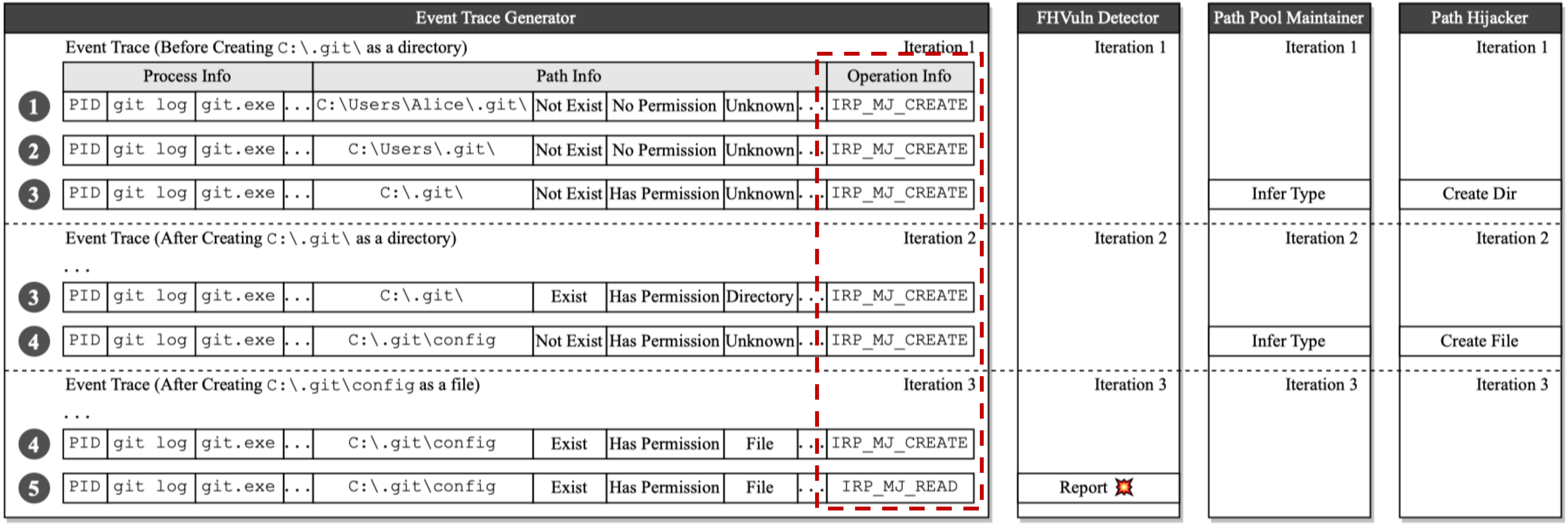
4	PID	git log	git.exe	. . .	C:\.git\config	Exist	Has Permission	File	. . .	IRP_MJ_CREATE
5	PID	git log	git.exe	. . .	C:\.git\config	Exist	Has Permission	File	. . .	IRP_MJ_READ

FHVuln Detector	Path Pool Maintainer	Path Hijacker
Iteration 1	Iteration 1 Infer Type	Iteration 1 Create Dir
Iteration 2	Iteration 2 Infer Type	Iteration 2 Create File
Iteration 3 Report 💥	Iteration 3	Iteration 3

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking

- **Event Trace Generator:** Execute the target program at each stage and records executed file operation traces
  - **Monitor Process Info:** process id、command line option (or Gui events)、executed program
  - **Monitor Path Info:** path、existence、permissions (whether can be manipulated by the path hijacker or not) 、type (file or directory)

Event Trace Generator



Event Trace Generator

Event Trace (Before Creating C:\.git\ as a directory) — Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory) — Iteration 2

. . .

	Process Info				Path Info					Operation Info
3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\config as a file) — Iteration 3

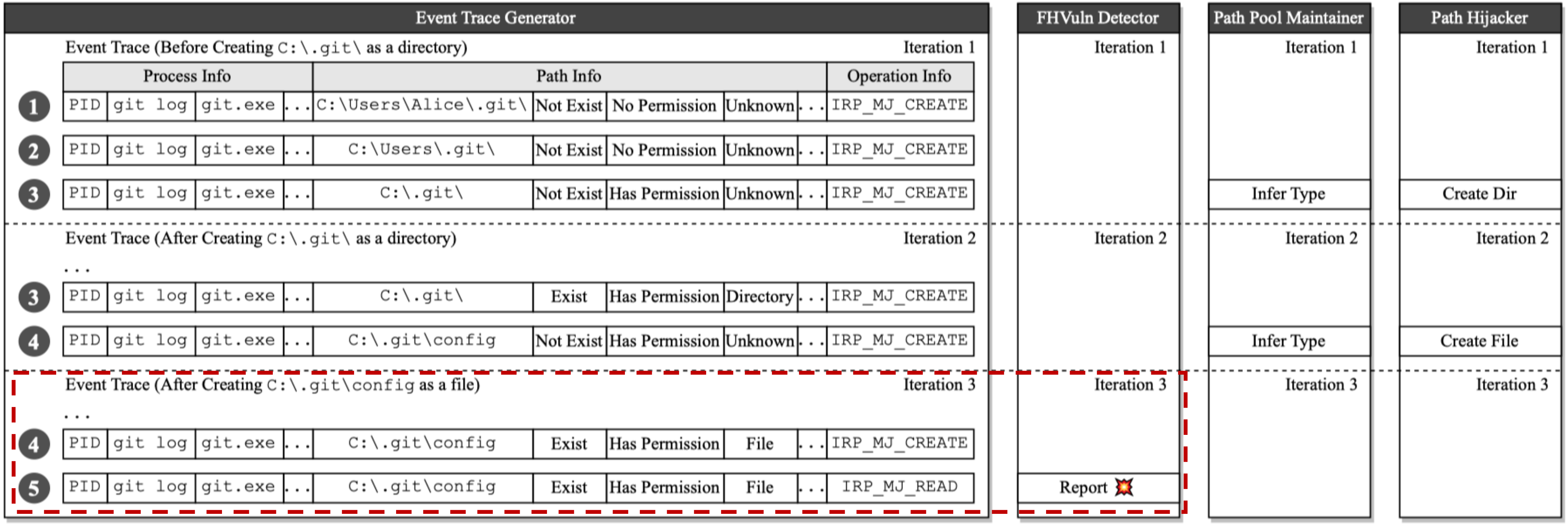
. . .

	Process Info				Path Info					Operation Info
4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking

- **Event Trace Generator:** Execute the target program at each stage and records executed file operation traces
  - **Monitor Process Info:** process id、command line option (or GUI events)、executed program
  - **Monitor Path Info:** path、 existence、permissions (whether can be manipulated by the path hijacker or not) 、type (file or directory)
  - **Monitor Operation:** moving、deleting、creating、reading…

FHVuln Detector



Event Trace Generator

Event Trace (Before Creating `C:\.git\` as a directory) Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating `C:\.git\` as a directory) Iteration 2

...

	Process Info				Path Info					Operation Info
3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating `C:\.git\config` as a file) Iteration 3

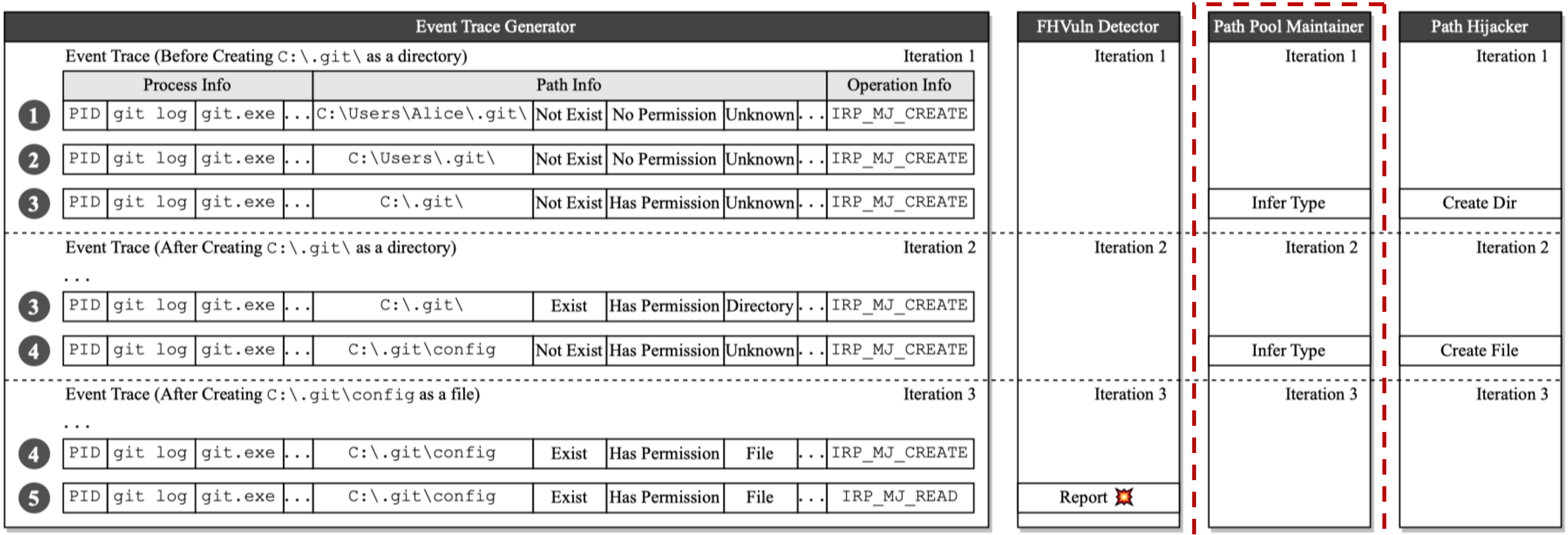
...

	Process Info				Path Info					Operation Info
4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking

- **FHVuln Detector:** Examine each execution traces and a FHVuln will be reported if the trace performs dangerous operations on hijacked files.
  - **hijacked file:** C:\.git\config
  - **dangerous operation:** reading

Path Pool Maintainer



Event Trace Generator

Event Trace (Before Creating C:\.git\ as a directory) Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory) Iteration 2

...

3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\config as a file) Iteration 3

...

4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

FHVuln Detector

Iteration 1

Iteration 2

Iteration 3

Report 💥

Path Pool Maintainer

Iteration 1

Infer Type

Iteration 2

Infer Type

Iteration 3

Path Hijacker

Iteration 1

Create Dir

Iteration 2

Create File

Iteration 3

(a) Event Trace Generation

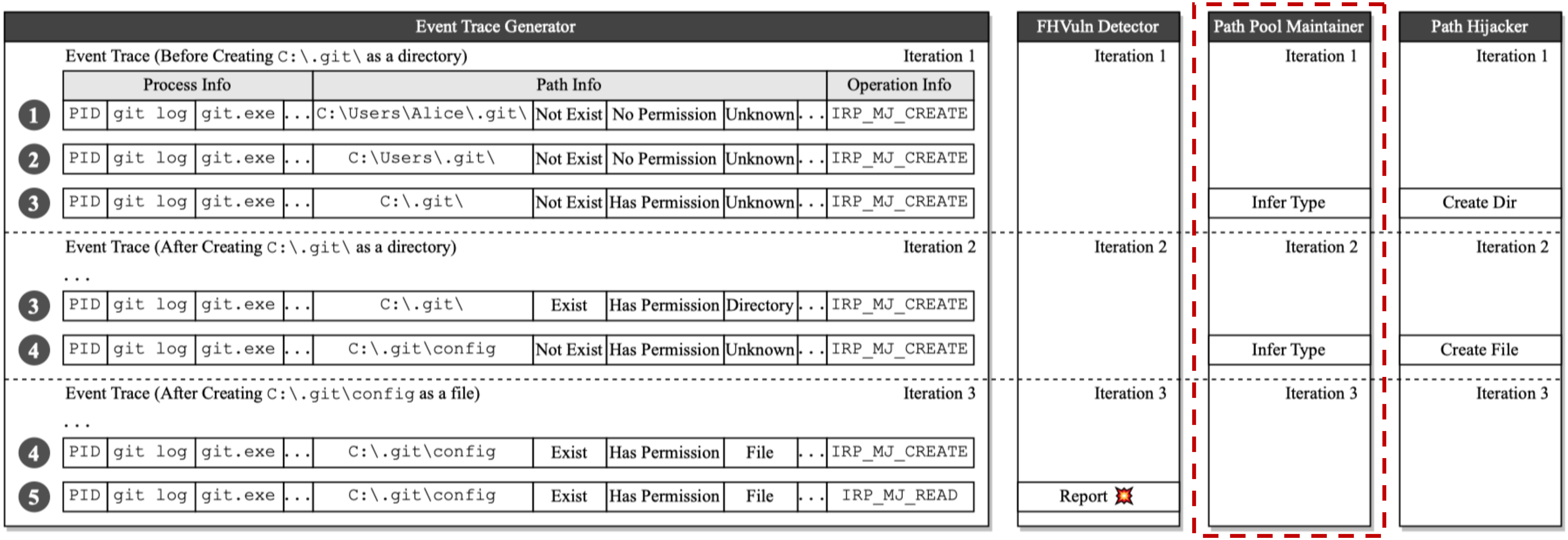
(b) FHVuln Detection

(c) Path Analysis

(d) Path Hijacking

- **Path Pool Maintainer:** Collect files encountered in the event trace and puts them into the path pool. In this step, JERRY also checks if the file refers to a normal file or a directory.

Path Pool Maintainer



Event Trace Generator

Event Trace (Before Creating C:\.git\ as a directory) — Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory) — Iteration 2

...

3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\config as a file) — Iteration 3

...

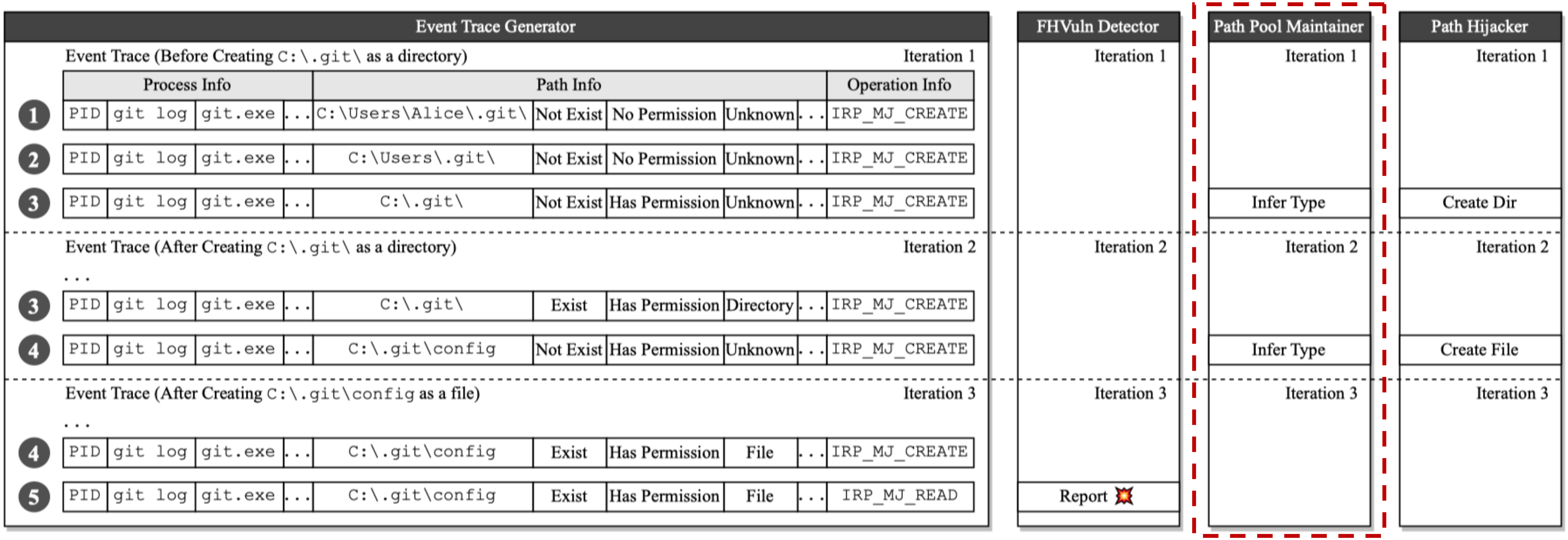
4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

	FHVuln Detector	Path Pool Maintainer	Path Hijacker
Iteration 1		Infer Type	Create Dir
Iteration 2		Infer Type	Create File
Iteration 3	Report 💥		

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking

- **Path Pool Maintainer:** Collect files encountered in the event trace and puts them into the path pool. In this step, JERRY also checks if the file refers to a normal file or a directory.
- **Heuristic:** When accessing a normal file, programs commonly check the existence of its parent directory while such a check is unnecessary when accessing a directory.

Path Pool Maintainer



Event Trace Generator

Event Trace (Before Creating C:\.git\ as a directory) Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory) Iteration 2

...

3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\config as a file) Iteration 3

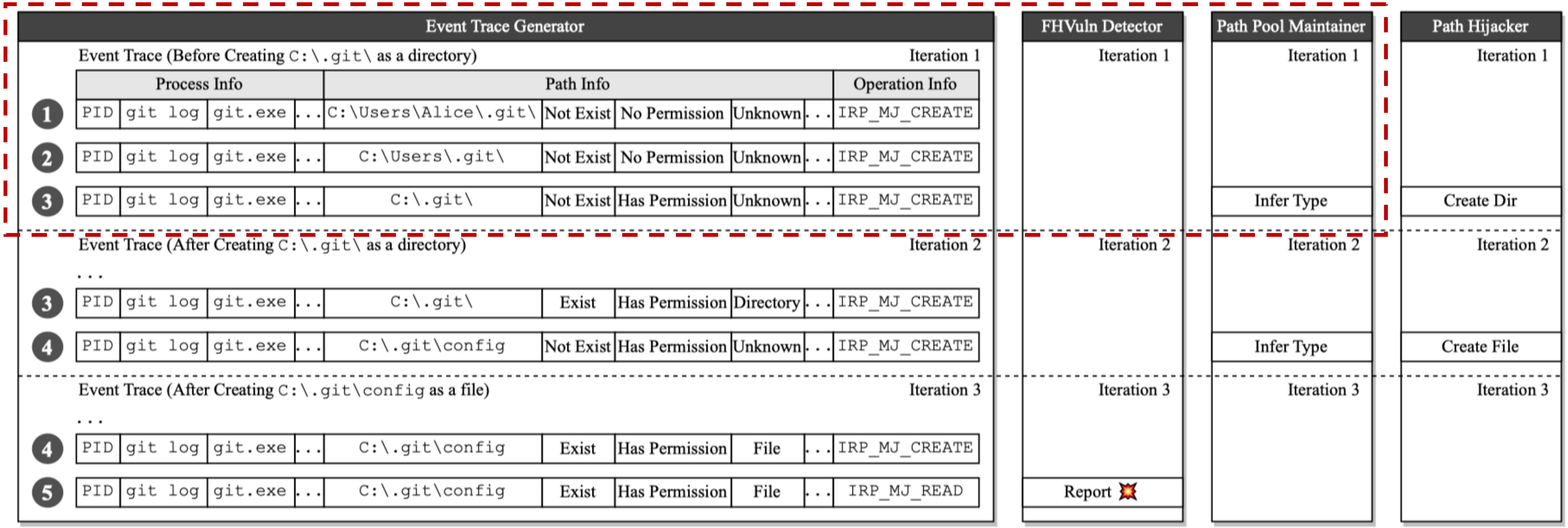
...

4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking

- **Path Pool Maintainer:** Collect files encountered in the event trace and puts them into the path pool. In this step, JERRY also checks if the file refers to a normal file or a directory.
- **Heuristic:** When accessing a normal file, programs commonly check the existence of its parent directory while such a check is unnecessary when accessing a directory.
- **Trial-and-error mechanism:** An encountered path with unknown type is by default considered as a file. If the file is used by file-specific operations later on, then the guess is correct. Otherwise, if the path is accessed by directory-specific operations, the guess is wrong and the path hijacker will create a directory instead.

Path Pool Maintainer



Event Trace (Before Creating C:\.git\ as a directory) Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory) Iteration 2

...

3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\config as a file) Iteration 3

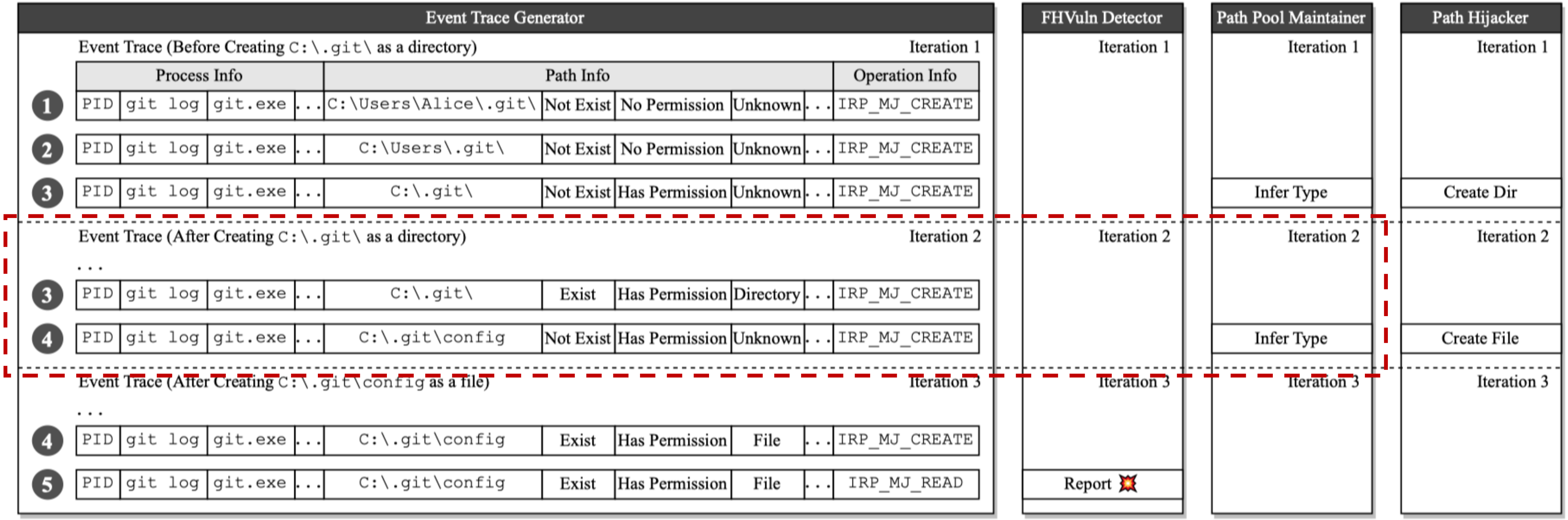
...

4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking

- **Iteration 1:**
  - **Guess:** C:\.git\ is a directory

Path Pool Maintainer



Event Trace (Before Creating C:\.git\ as a directory)

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory)

...

3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\config as a file)

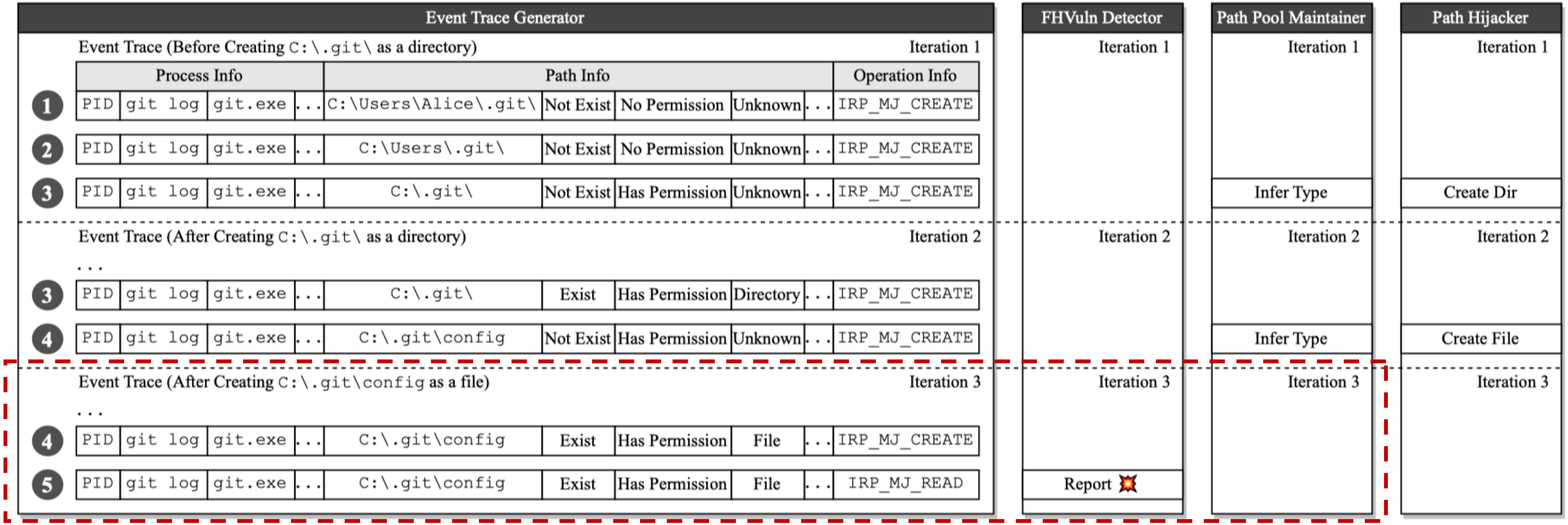
...

4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking

- **Iteration 1:**
  - **Guess:** C:\.git\ is a directory
- **Iteration 2:**
  - **Check:** directory-specific operation on C:\.git\, so C:\.git is a directory
  - **Guess:** C:\.git\config is a file

Path Pool Maintainer



Event Trace Generator

Event Trace (Before Creating C:\.git\ as a directory) Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory) Iteration 2

. . .

3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\config as a file) Iteration 3

. . .

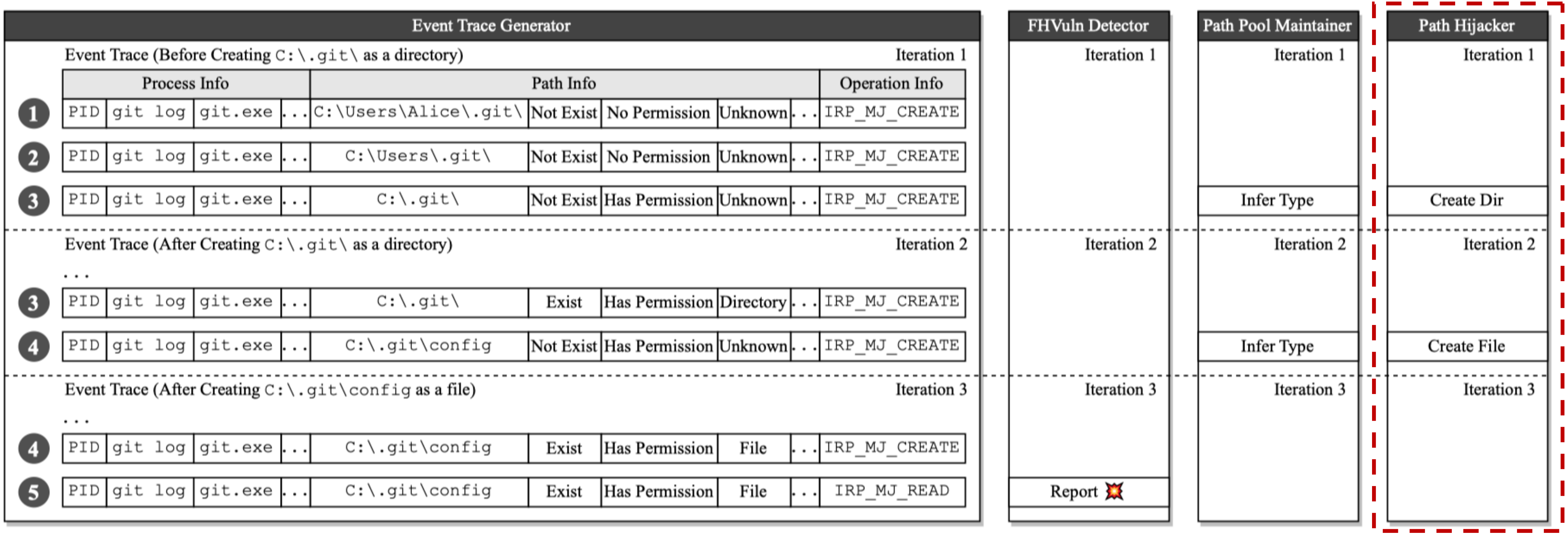
4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

FHVuln Detector	Path Pool Maintainer	Path Hijacker
Iteration 1	Iteration 1	Iteration 1
	Infer Type	Create Dir
Iteration 2	Iteration 2	Iteration 2
	Infer Type	Create File
Iteration 3	Iteration 3	Iteration 3
Report 💥		

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking

- **Iteration 1:**
  - **Guess:** C:\.git\ is a directory
- **Iteration 2:**
  - **Check:** directory-specific operation on C:\.git\, so C:\.git is a directory
  - **Guess:** C:\.git\config is a file
- **Iteration 3:**
  - **Check:** file-specific operation on C:\.git\config, so C:\.git\config is a directory

Path Hijacker



Event Trace Generator

Event Trace (Before Creating C:\.git\ as a directory) Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory) Iteration 2

...

3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\config as a file) Iteration 3

...

4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

FHVuln Detector

Iteration 1

Iteration 2

Iteration 3

Report 💥

Path Pool Maintainer

Iteration 1

Infer Type

Iteration 2

Infer Type

Iteration 3

Path Hijacker

Iteration 1

Create Dir

Iteration 2

Create File

Iteration 3

(a) Event Trace Generation

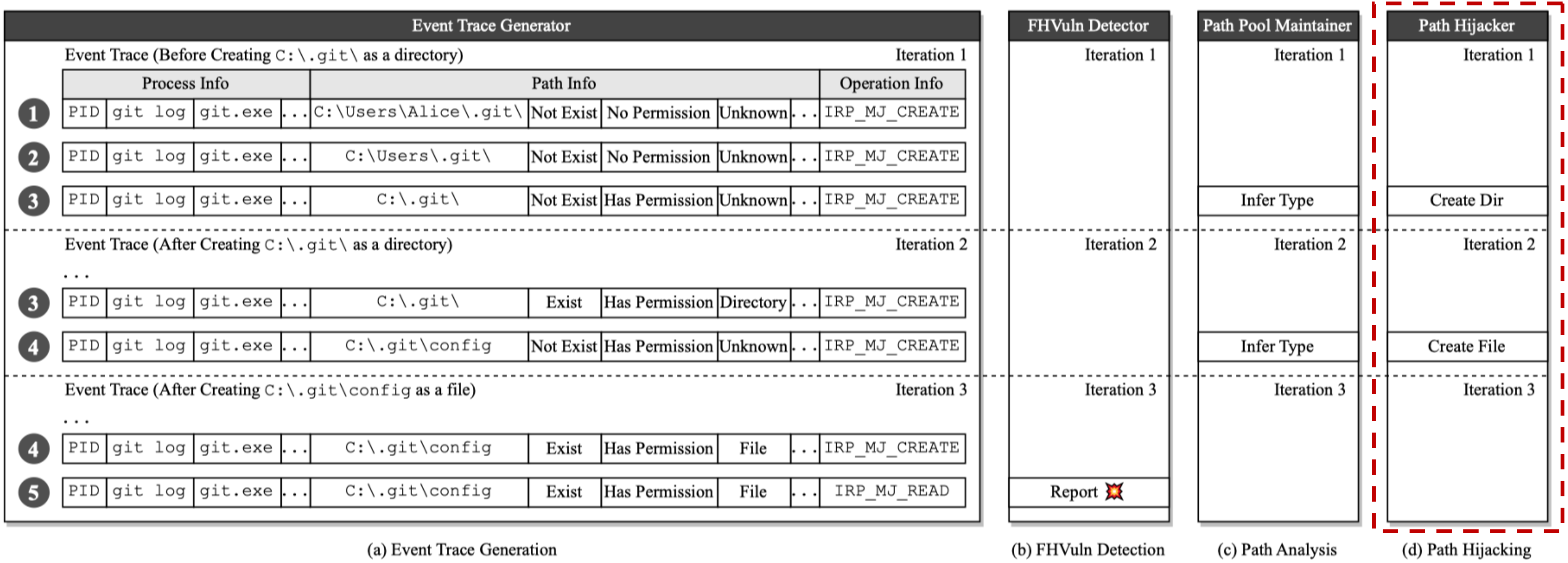
(b) FHVuln Detection

(c) Path Analysis

(d) Path Hijacking

- **Path Hijacker:** Hijack file or file path as an attacker

Path Hijacker



Event Trace (Before Creating `C:\.git\` as a directory) — Iteration 1

	Process Info				Path Info					Operation Info
1	`PID`	`git log`	`git.exe`	...	`C:\Users\Alice\.git\`	Not Exist	No Permission	Unknown	...	`IRP_MJ_CREATE`
2	`PID`	`git log`	`git.exe`	...	`C:\Users\.git\`	Not Exist	No Permission	Unknown	...	`IRP_MJ_CREATE`
3	`PID`	`git log`	`git.exe`	...	`C:\.git\`	Not Exist	Has Permission	Unknown	...	`IRP_MJ_CREATE`

Event Trace (After Creating `C:\.git\` as a directory) — Iteration 2

...

3	`PID`	`git log`	`git.exe`	...	`C:\.git\`	Exist	Has Permission	Directory	...	`IRP_MJ_CREATE`
4	`PID`	`git log`	`git.exe`	...	`C:\.git\config`	Not Exist	Has Permission	Unknown	...	`IRP_MJ_CREATE`

Event Trace (After Creating `C:\.git\config` as a file) — Iteration 3

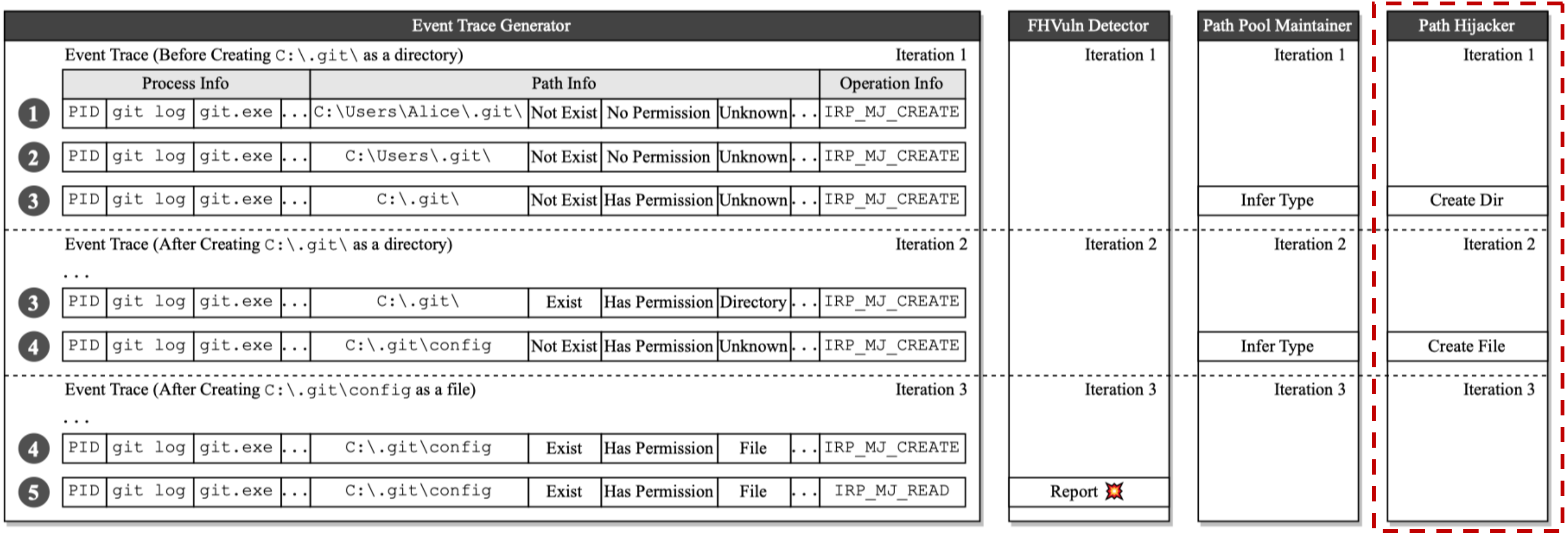
...

4	`PID`	`git log`	`git.exe`	...	`C:\.git\config`	Exist	Has Permission	File	...	`IRP_MJ_CREATE`
5	`PID`	`git log`	`git.exe`	...	`C:\.git\config`	Exist	Has Permission	File	...	`IRP_MJ_READ`

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking

- **Path Hijacker:** Hijack file or file path as an attacker
  - **For exe and dll:** replace with manually crafted files

Path Hijacker



Event Trace Generator

Event Trace (Before Creating C:\.git\ as a directory) Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\ as a directory) Iteration 2

...

3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating C:\.git\config as a file) Iteration 3

...

4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

FHVuln Detector

Iteration 1

Iteration 2

Iteration 3

Report 💥

Path Pool Maintainer

Iteration 1

Infer Type

Iteration 2

Infer Type

Iteration 3

Path Hijacker

Iteration 1

Create Dir

Iteration 2

Create File

Iteration 3

(a) Event Trace Generation

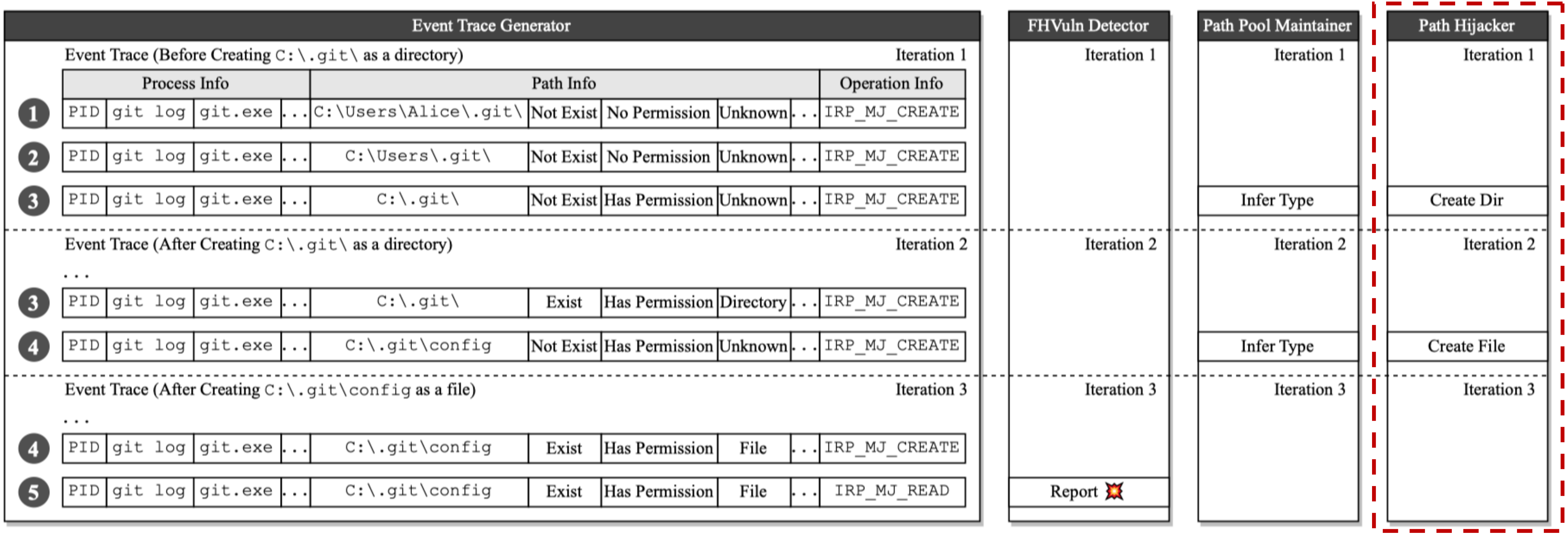
(b) FHVuln Detection

(c) Path Analysis

(d) Path Hijacking

- **Path Hijacker:** Hijack file or file path as an attacker
  - **For exe and dll:** replace with manually crafted files
  - **For other files:** a specially created blank file consisting of newline and space characters

Path Hijacker



Event Trace (Before Creating `C:\.git\` as a directory) — Iteration 1

	Process Info				Path Info					Operation Info
1	PID	git log	git.exe	...	C:\Users\Alice\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
2	PID	git log	git.exe	...	C:\Users\.git\	Not Exist	No Permission	Unknown	...	IRP_MJ_CREATE
3	PID	git log	git.exe	...	C:\.git\	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating `C:\.git\` as a directory) — Iteration 2

...

3	PID	git log	git.exe	...	C:\.git\	Exist	Has Permission	Directory	...	IRP_MJ_CREATE
4	PID	git log	git.exe	...	C:\.git\config	Not Exist	Has Permission	Unknown	...	IRP_MJ_CREATE

Event Trace (After Creating `C:\.git\config` as a file) — Iteration 3

...

4	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_CREATE
5	PID	git log	git.exe	...	C:\.git\config	Exist	Has Permission	File	...	IRP_MJ_READ

(a) Event Trace Generation (b) FHVuln Detection (c) Path Analysis (d) Path Hijacking

- **Path Hijacker:** Hijack file or file path as an attacker
  - **For exe and dll:** replace with manually crafted files.
  - **For other files:** a specially created blank file consisting of newline and space characters
  - **For creating、moving、deleting operations:** Create a symbolic link pointing to a special location for monitoring

Experiment Setup

- Two Benchmark

Known Benchmark
51 FHVulns
from Study

Unkown Benchmark
438 Real-world
Software

Experiment Setup

- Two Benchmark

Known Benchmark
51 FHVulns
from Study

Unkown Benchmark
438 Real-world
Software

- Baseline:
  - PrivescCheck: A static tool analyses access control list of file (directory)
  - JERRY-Crassus: Crassus is a FHVuln detection tool by analysing event traces captured by ProcMon. We extended Crassus by incorporating the event trace generator module and replaced our monitor with ProcMon
  - JERRY

Experiment Setup

- Two Benchmark

Known Benchmark
51 FHVulns
from Study

Unkown Benchmark
438 Real-world
Software

- Baseline:
  - PrivescCheck: A static tool analyses access control list of file (directory)
  - JERRY-Crassus: Crassus is a FHVuln detection tool by analysing event traces captured by ProcMon. We extended Crassus by incorporating the event trace generator module and replaced our monitor with ProcMon
  - JERRY

- Three Experiments:
  - **Effectiveness on Known Vulnerabilities**
  - **Effectiveness on Unknown Vulnerabilities**
  - **Efficiency**

Effectiveness on Known Vulnerabilities

Tool	# reported	TP	FP	FN	Precision	Recall
PrivescCheck	34	20	14	31	58.8%	39.2%
JERRY-Crassus	44	37	7	14	84.1%	72.5%
JERRY	**50**	**50**	**0**	**1**	**100%**	**98.0%**

FN of JERRY：Complex use situation

Effectiveness on Known Vulnerabilities

Tool	# reported	TP	FP	FN	Precision	Recall
PrivescCheck	34	20	14	31	58.8%	39.2%
JERRY-Crassus	44	37	7	14	84.1%	72.5%
JERRY	**50**	**50**	**0**	**1**	**100%**	**98.0%**

FN of JERRY：Complex use situation

FP of PrivescCheck：Scanning the parent directory permissions when encountering an executable. (e.g. dll hijacking)
FP of JERRY-Crassus：hijacking does not means the file will be used by program

Effectiveness on Known Vulnerabilities

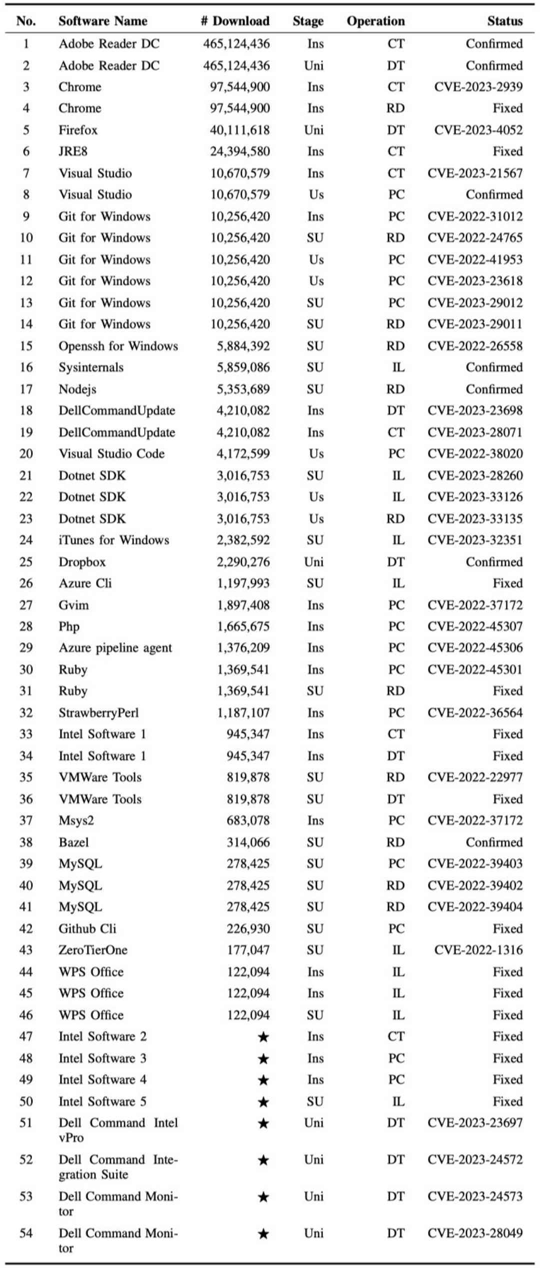
Tool	# reported	TP	FP	FN	Precision	Recall
PrivescCheck	34	20	14	31	58.8%	39.2%
JERRY-Crassus	44	37	7	14	84.1%	72.5%
JERRY	**50**	**50**	**0**	**1**	**100%**	**98.0%**

FN of JERRY：Complex use situation

FP of PrivescCheck：Scanning the parent directory permissions when encountering an executable. (e.g. dll hijacking)
FP of JERRY-Crassus：hijacking does not means the file will be used by program

FN of PrivescCheck：1. uncomplete sensitive operations (reading (10), creating (5), deleting (4), and moving (1)); 2. only consider after installation (9); 3. others (2)
FN of JERRY-Crassus：1. uncomplete sensitive operations (creating (5), deleting (4), and moving (1)); 2. only consider reading related to openssl.cnf (3)

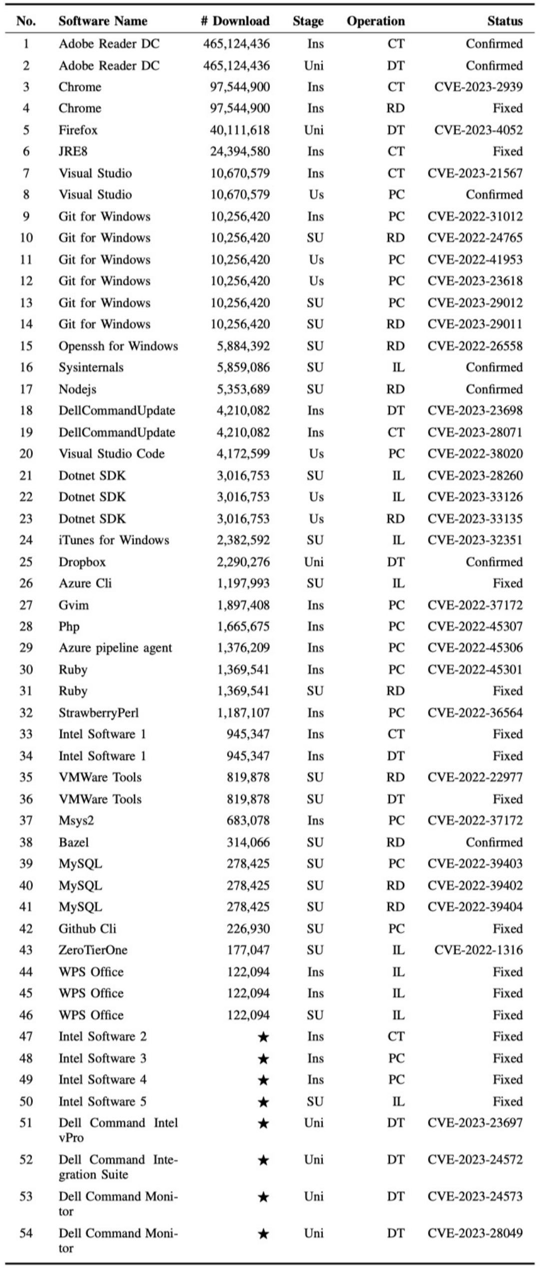
Effectiveness on Unknown Vulnerabilities



No.	Software Name	# Download	Stage	Operation	Status
1	Adobe Reader DC	465,124,436	Ins	CT	Confirmed
2	Adobe Reader DC	465,124,436	Uni	DT	Confirmed
3	Chrome	97,544,900	Ins	CT	CVE-2023-2939
4	Chrome	97,544,900	Ins	RD	Fixed
5	Firefox	40,111,618	Uni	DT	CVE-2023-4052
6	JRE8	24,394,580	Ins	CT	Fixed
7	Visual Studio	10,670,579	Ins	CT	CVE-2023-21567
8	Visual Studio	10,670,579	Us	PC	Confirmed
9	Git for Windows	10,256,420	Ins	PC	CVE-2022-31012
10	Git for Windows	10,256,420	SU	RD	CVE-2022-24765
11	Git for Windows	10,256,420	Us	PC	CVE-2022-41953
12	Git for Windows	10,256,420	Us	PC	CVE-2023-23618
13	Git for Windows	10,256,420	SU	PC	CVE-2023-29012
14	Git for Windows	10,256,420	SU	RD	CVE-2023-29011
15	Openssh for Windows	5,884,392	SU	RD	CVE-2022-26558
16	Sysinternals	5,859,086	SU	IL	Confirmed
17	Nodejs	5,353,689	SU	RD	Confirmed
18	DellCommandUpdate	4,210,082	Ins	DT	CVE-2023-23698
19	DellCommandUpdate	4,210,082	Ins	CT	CVE-2023-28071
20	Visual Studio Code	4,172,599	Us	PC	CVE-2022-38020
21	Dotnet SDK	3,016,753	SU	IL	CVE-2023-28260
22	Dotnet SDK	3,016,753	Us	IL	CVE-2023-33126
23	Dotnet SDK	3,016,753	Us	RD	CVE-2023-33135
24	iTunes for Windows	2,382,592	SU	IL	CVE-2023-32351
25	Dropbox	2,290,276	Uni	DT	Confirmed
26	Azure Cli	1,197,993	SU	IL	Fixed
27	Gvim	1,897,408	Ins	PC	CVE-2022-37172
28	Php	1,665,675	Ins	PC	CVE-2022-45307
29	Azure pipeline agent	1,376,209	Ins	PC	CVE-2022-45306
30	Ruby	1,369,541	Ins	PC	CVE-2022-45301
31	Ruby	1,369,541	SU	RD	Fixed
32	StrawberryPerl	1,187,107	Ins	PC	CVE-2022-36564
33	Intel Software 1	945,347	Ins	CT	Fixed
34	Intel Software 1	945,347	Ins	DT	Fixed
35	VMWare Tools	819,878	SU	RD	CVE-2022-22977
36	VMWare Tools	819,878	SU	DT	Fixed
37	Msys2	683,078	Ins	PC	CVE-2022-37172
38	Bazel	314,066	SU	RD	Confirmed
39	MySQL	278,425	SU	PC	CVE-2022-39403
40	MySQL	278,425	SU	RD	CVE-2022-39402
41	MySQL	278,425	SU	RD	CVE-2022-39404
42	Github Cli	226,930	SU	PC	Fixed
43	ZeroTierOne	177,047	SU	IL	CVE-2022-1316
44	WPS Office	122,094	Ins	IL	Fixed
45	WPS Office	122,094	Ins	IL	Fixed
46	WPS Office	122,094	SU	IL	Fixed
47	Intel Software 2	★	Ins	CT	Fixed
48	Intel Software 3	★	Ins	PC	Fixed
49	Intel Software 4	★	Ins	PC	Fixed
50	Intel Software 5	★	SU	IL	Fixed
51	Dell Command Intel vPro	★	Uni	DT	CVE-2023-23697
52	Dell Command Integration Suite	★	Uni	DT	CVE-2023-24572
53	Dell Command Monitor	★	Uni	DT	CVE-2023-24573
54	Dell Command Monitor	★	Uni	DT	CVE-2023-28049

JERRY Find 339 zero-day FHVulns in 438 Real-world software with 21 false positive.

Effectiveness on Unknown Vulnerabilities

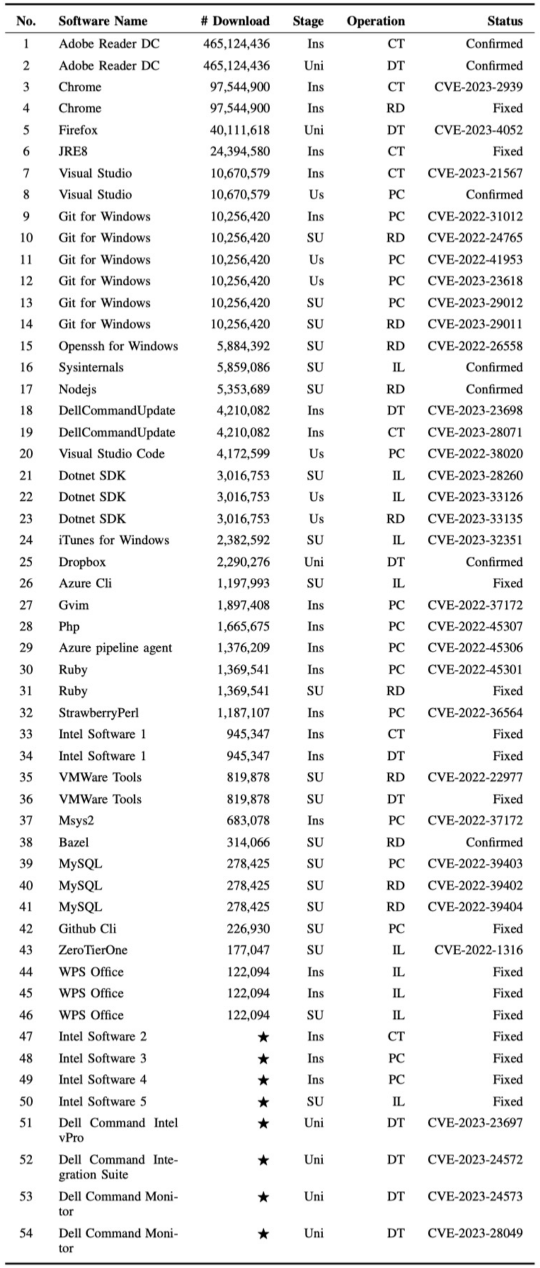


No.	Software Name	# Download	Stage	Operation	Status
1	Adobe Reader DC	465,124,436	Ins	CT	Confirmed
2	Adobe Reader DC	465,124,436	Uni	DT	Confirmed
3	Chrome	97,544,900	Ins	CT	CVE-2023-2939
4	Chrome	97,544,900	Ins	RD	Fixed
5	Firefox	40,111,618	Uni	DT	CVE-2023-4052
6	JRE8	24,394,580	Ins	CT	Fixed
7	Visual Studio	10,670,579	Ins	CT	CVE-2023-21567
8	Visual Studio	10,670,579	Us	PC	Confirmed
9	Git for Windows	10,256,420	Ins	PC	CVE-2022-31012
10	Git for Windows	10,256,420	SU	RD	CVE-2022-24765
11	Git for Windows	10,256,420	Us	PC	CVE-2022-41953
12	Git for Windows	10,256,420	Us	PC	CVE-2023-23618
13	Git for Windows	10,256,420	SU	PC	CVE-2023-29012
14	Git for Windows	10,256,420	SU	RD	CVE-2023-29011
15	Openssh for Windows	5,884,392	SU	RD	CVE-2022-26558
16	Sysinternals	5,859,086	SU	IL	Confirmed
17	Nodejs	5,353,689	SU	RD	Confirmed
18	DellCommandUpdate	4,210,082	Ins	DT	CVE-2023-23698
19	DellCommandUpdate	4,210,082	Ins	CT	CVE-2023-28071
20	Visual Studio Code	4,172,599	Us	PC	CVE-2022-38020
21	Dotnet SDK	3,016,753	SU	IL	CVE-2023-28260
22	Dotnet SDK	3,016,753	Us	IL	CVE-2023-33126
23	Dotnet SDK	3,016,753	Us	RD	CVE-2023-33135
24	iTunes for Windows	2,382,592	SU	IL	CVE-2023-32351
25	Dropbox	2,290,276	Uni	DT	Confirmed
26	Azure Cli	1,197,993	SU	IL	Fixed
27	Gvim	1,897,408	Ins	PC	CVE-2022-37172
28	Php	1,665,675	Ins	PC	CVE-2022-45307
29	Azure pipeline agent	1,376,209	Ins	PC	CVE-2022-45306
30	Ruby	1,369,541	Ins	PC	CVE-2022-45301
31	Ruby	1,369,541	SU	RD	Fixed
32	StrawberryPerl	1,187,107	Ins	PC	CVE-2022-36564
33	Intel Software 1	945,347	Ins	CT	Fixed
34	Intel Software 1	945,347	Ins	DT	Fixed
35	VMWare Tools	819,878	SU	RD	CVE-2022-22977
36	VMWare Tools	819,878	SU	DT	Fixed
37	Msys2	683,078	Ins	PC	CVE-2022-37172
38	Bazel	314,066	SU	RD	Confirmed
39	MySQL	278,425	SU	PC	CVE-2022-39403
40	MySQL	278,425	SU	RD	CVE-2022-39402
41	MySQL	278,425	SU	RD	CVE-2022-39404
42	Github Cli	226,930	SU	PC	Fixed
43	ZeroTierOne	177,047	SU	IL	CVE-2022-1316
44	WPS Office	122,094	Ins	IL	Fixed
45	WPS Office	122,094	Ins	IL	Fixed
46	WPS Office	122,094	SU	IL	Fixed
47	Intel Software 2	★	Ins	CT	Fixed
48	Intel Software 3	★	Ins	PC	Fixed
49	Intel Software 4	★	Ins	PC	Fixed
50	Intel Software 5	★	SU	IL	Fixed
51	Dell Command Intel vPro	★	Uni	DT	CVE-2023-23697
52	Dell Command Integration Suite	★	Uni	DT	CVE-2023-24572
53	Dell Command Monitor	★	Uni	DT	CVE-2023-24573
54	Dell Command Monitor	★	Uni	DT	CVE-2023-28049

JERRY Find 339 zero-day FHVulns in 438 Real-world software with 21 false positive.

PrivesCheck only found 39 FHVulns (11.5% of JERRY)

Effectiveness on Unknown Vulnerabilities



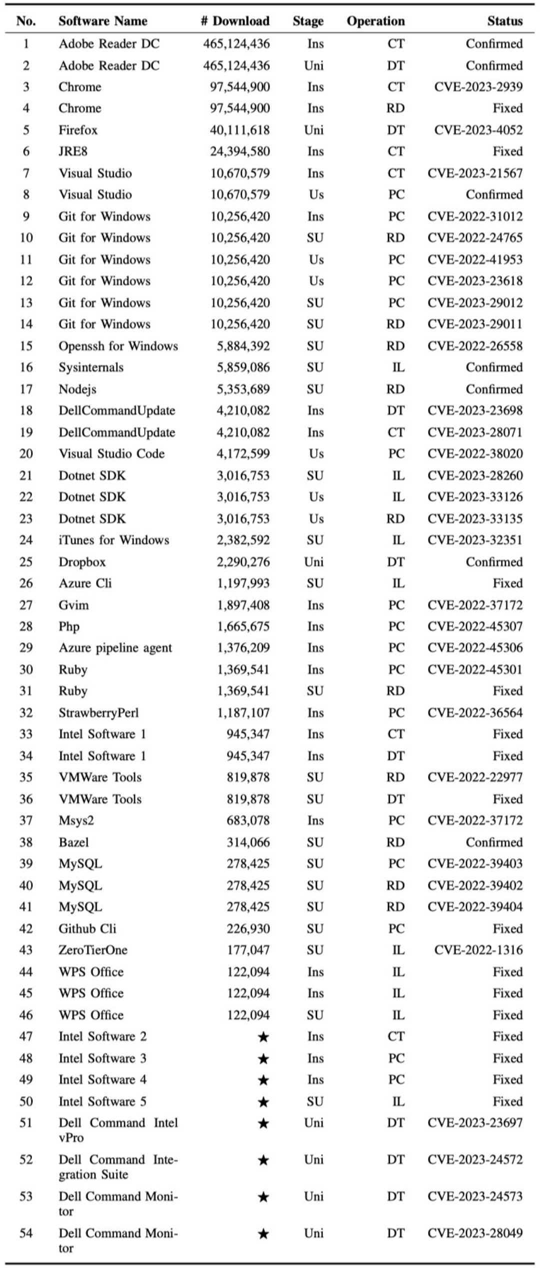
No.	Software Name	# Download	Stage	Operation	Status
1	Adobe Reader DC	465,124,436	Ins	CT	Confirmed
2	Adobe Reader DC	465,124,436	Uni	DT	Confirmed
3	Chrome	97,544,900	Ins	CT	CVE-2023-2939
4	Chrome	97,544,900	Ins	RD	Fixed
5	Firefox	40,111,618	Uni	DT	CVE-2023-4052
6	JRE8	24,394,580	Ins	CT	Fixed
7	Visual Studio	10,670,579	Ins	CT	CVE-2023-21567
8	Visual Studio	10,670,579	Us	PC	Confirmed
9	Git for Windows	10,256,420	Ins	PC	CVE-2022-31012
10	Git for Windows	10,256,420	SU	RD	CVE-2022-24765
11	Git for Windows	10,256,420	Us	PC	CVE-2022-41953
12	Git for Windows	10,256,420	Us	PC	CVE-2023-23618
13	Git for Windows	10,256,420	SU	PC	CVE-2023-29012
14	Git for Windows	10,256,420	SU	RD	CVE-2023-29011
15	Openssh for Windows	5,884,392	SU	RD	CVE-2022-26558
16	Sysinternals	5,859,086	SU	IL	Confirmed
17	Nodejs	5,353,689	SU	RD	Confirmed
18	DellCommandUpdate	4,210,082	Ins	DT	CVE-2023-23698
19	DellCommandUpdate	4,210,082	Ins	CT	CVE-2023-28071
20	Visual Studio Code	4,172,599	Us	PC	CVE-2022-38020
21	Dotnet SDK	3,016,753	SU	IL	CVE-2023-28260
22	Dotnet SDK	3,016,753	Us	IL	CVE-2023-33126
23	Dotnet SDK	3,016,753	Us	RD	CVE-2023-33135
24	iTunes for Windows	2,382,592	SU	IL	CVE-2023-32351
25	Dropbox	2,290,276	Uni	DT	Confirmed
26	Azure Cli	1,197,993	SU	IL	Fixed
27	Gvim	1,897,408	Ins	PC	CVE-2022-37172
28	Php	1,665,675	Ins	PC	CVE-2022-45307
29	Azure pipeline agent	1,376,209	Ins	PC	CVE-2022-45306
30	Ruby	1,369,541	Ins	PC	CVE-2022-45301
31	Ruby	1,369,541	SU	RD	Fixed
32	StrawberryPerl	1,187,107	Ins	PC	CVE-2022-36564
33	Intel Software 1	945,347	Ins	CT	Fixed
34	Intel Software 1	945,347	Ins	DT	Fixed
35	VMWare Tools	819,878	SU	RD	CVE-2022-22977
36	VMWare Tools	819,878	SU	DT	Fixed
37	Msys2	683,078	Ins	PC	CVE-2022-37172
38	Bazel	314,066	SU	RD	Confirmed
39	MySQL	278,425	SU	PC	CVE-2022-39403
40	MySQL	278,425	SU	RD	CVE-2022-39402
41	MySQL	278,425	SU	RD	CVE-2022-39404
42	Github Cli	226,930	SU	PC	Fixed
43	ZeroTierOne	177,047	SU	IL	CVE-2022-1316
44	WPS Office	122,094	Ins	IL	Fixed
45	WPS Office	122,094	Ins	IL	Fixed
46	WPS Office	122,094	SU	IL	Fixed
47	Intel Software 2	★	Ins	CT	Fixed
48	Intel Software 3	★	Ins	PC	Fixed
49	Intel Software 4	★	Ins	PC	Fixed
50	Intel Software 5	★	SU	IL	Fixed
51	Dell Command Intel vPro	★	Uni	DT	CVE-2023-23697
52	Dell Command Integration Suite	★	Uni	DT	CVE-2023-24572
53	Dell Command Monitor	★	Uni	DT	CVE-2023-24573
54	Dell Command Monitor	★	Uni	DT	CVE-2023-28049

JERRY Find 339 zero-day FHVulns in 438 Real-world software with 21 false positive.

PrivesCheck only found 39 FHVulns (11.5% of JERRY)

JERRY-Crassus detect 143 FHVulns (42.2% of JERRY)

Effectiveness on Unknown Vulnerabilities



No.	Software Name	# Download	Stage	Operation	Status
1	Adobe Reader DC	465,124,436	Ins	CT	Confirmed
2	Adobe Reader DC	465,124,436	Uni	DT	Confirmed
3	Chrome	97,544,900	Ins	CT	CVE-2023-2939
4	Chrome	97,544,900	Ins	RD	Fixed
5	Firefox	40,111,618	Uni	DT	CVE-2023-4052
6	JRE8	24,394,580	Ins	CT	Fixed
7	Visual Studio	10,670,579	Ins	CT	CVE-2023-21567
8	Visual Studio	10,670,579	Us	PC	Confirmed
9	Git for Windows	10,256,420	Ins	PC	CVE-2022-31012
10	Git for Windows	10,256,420	SU	RD	CVE-2022-24765
11	Git for Windows	10,256,420	Us	PC	CVE-2022-41953
12	Git for Windows	10,256,420	Us	PC	CVE-2023-23618
13	Git for Windows	10,256,420	SU	PC	CVE-2023-29012
14	Git for Windows	10,256,420	SU	RD	CVE-2023-29011
15	Openssh for Windows	5,884,392	SU	RD	CVE-2022-26558
16	Sysinternals	5,859,086	SU	IL	Confirmed
17	Nodejs	5,353,689	SU	RD	Confirmed
18	DellCommandUpdate	4,210,082	Ins	DT	CVE-2023-23698
19	DellCommandUpdate	4,210,082	Ins	CT	CVE-2023-28071
20	Visual Studio Code	4,172,599	Us	PC	CVE-2022-38020
21	Dotnet SDK	3,016,753	SU	IL	CVE-2023-28260
22	Dotnet SDK	3,016,753	Us	IL	CVE-2023-33126
23	Dotnet SDK	3,016,753	Us	RD	CVE-2023-33135
24	iTunes for Windows	2,382,592	SU	IL	CVE-2023-32351
25	Dropbox	2,290,276	Uni	DT	Confirmed
26	Azure Cli	1,197,993	SU	IL	Fixed
27	Gvim	1,897,408	Ins	PC	CVE-2022-37172
28	Php	1,665,675	Ins	PC	CVE-2022-45307
29	Azure pipeline agent	1,376,209	Ins	PC	CVE-2022-45306
30	Ruby	1,369,541	Ins	PC	CVE-2022-45301
31	Ruby	1,369,541	SU	RD	Fixed
32	StrawberryPerl	1,187,107	Ins	PC	CVE-2022-36564
33	Intel Software 1	945,347	Ins	CT	Fixed
34	Intel Software 1	945,347	Ins	DT	Fixed
35	VMWare Tools	819,878	SU	RD	CVE-2022-22977
36	VMWare Tools	819,878	SU	DT	Fixed
37	Msys2	683,078	Ins	PC	CVE-2022-37172
38	Bazel	314,066	SU	RD	Confirmed
39	MySQL	278,425	SU	PC	CVE-2022-39403
40	MySQL	278,425	SU	RD	CVE-2022-39402
41	MySQL	278,425	SU	RD	CVE-2022-39404
42	Github Cli	226,930	SU	PC	Fixed
43	ZeroTierOne	177,047	SU	IL	CVE-2022-1316
44	WPS Office	122,094	Ins	IL	Fixed
45	WPS Office	122,094	Ins	IL	Fixed
46	WPS Office	122,094	SU	IL	Fixed
47	Intel Software 2	★	Ins	CT	Fixed
48	Intel Software 3	★	Ins	PC	Fixed
49	Intel Software 4	★	Ins	PC	Fixed
50	Intel Software 5	★	SU	IL	Fixed
51	Dell Command Intel vPro	★	Uni	DT	CVE-2023-23697
52	Dell Command Integration Suite	★	Uni	DT	CVE-2023-24572
53	Dell Command Monitor	★	Uni	DT	CVE-2023-24573
54	Dell Command Monitor	★	Uni	DT	CVE-2023-28049

JERRY Find 339 zero-day FHVulns in 438 Real-world software with 21 false positive.

PrivesCheck only found 39 FHVulns (11.5% of JERRY)

JERRY-Crassus detect 143 FHVulns (42.2% of JERRY)

FP Analysis：All FP issues are related to read operation.

Type 1: Read but not actually used by program

Type 2: Read but can not exploit in Windows system

Efficiency

Tool	Install	Unistall	Update	Repair	StartUp	Usage
JERRY-NoInfer	8039.4	1417.6	3871.9	1206.9	556.2	33.8
JERRY	**1128.1**	**414.2**	**893.5**	**254.7**	**115.5**	**15.8**

JERRY-NoInfer：does not use our proposed path type inference and tested these paths which cannot decide whether file or directory by our heuristics directly one by one.

Efficiency

Tool	Install	Unistall	Update	Repair	StartUp	Usage
JERRY-NoInfer	8039.4	1417.6	3871.9	1206.9	556.2	33.8
JERRY	**1128.1**	**414.2**	**893.5**	**254.7**	**115.5**	**15.8**

JERRY-NoInfer：does not use our proposed path type inference and tested these paths which cannot decide whether file or directory by our heuristics directly one by one.

JERRY achieved at least 2.14 faster in the usage stage and 7.13 faster in the installation stage

Efficiency

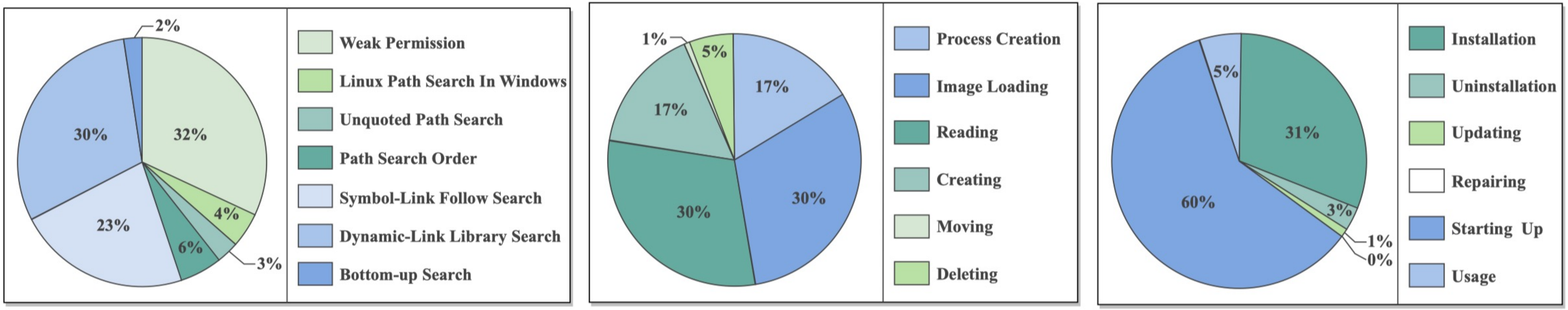
Tool	Install	Unistall	Update	Repair	StartUp	Usage
JERRY-NoInfer	8039.4	1417.6	3871.9	1206.9	556.2	33.8
JERRY	**1128.1**	**414.2**	**893.5**	**254.7**	**115.5**	**15.8**

JERRY-NoInfer：does not use our proposed path type inference and tested these paths which cannot decide whether file or directory by our heuristics directly one by one.

JERRY achieved at least 2.14 faster in the usage stage and 7.13 faster in the installation stage

Only a few paths that can be hijacked in the usage stage
In the installation stage, there are much more paths that can be hijacked than other stages.

Analysis of New FHVulns



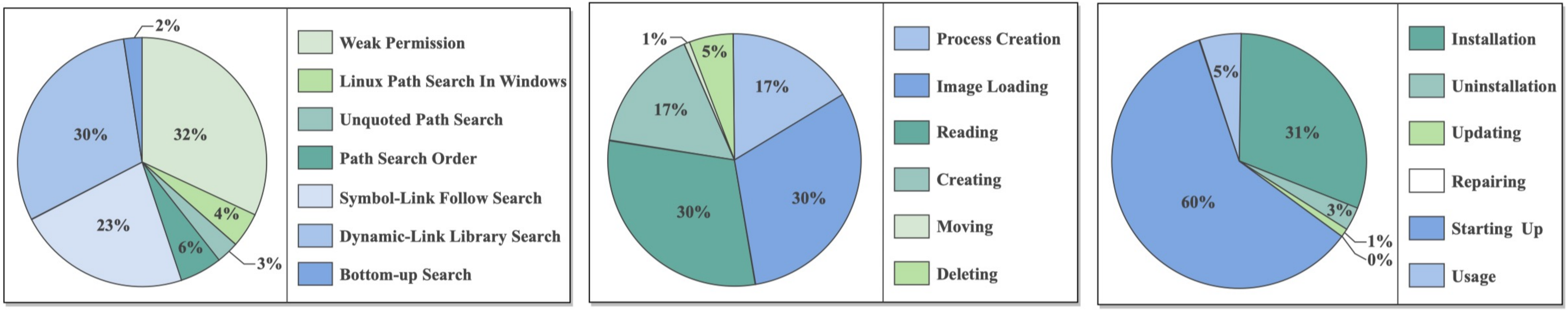
(a) Origins

(b) File Operations

(c) Stages

Distribution of new FHVulns on different origins, file operations, and stages.

Analysis of New FHVulns



(a) Origins

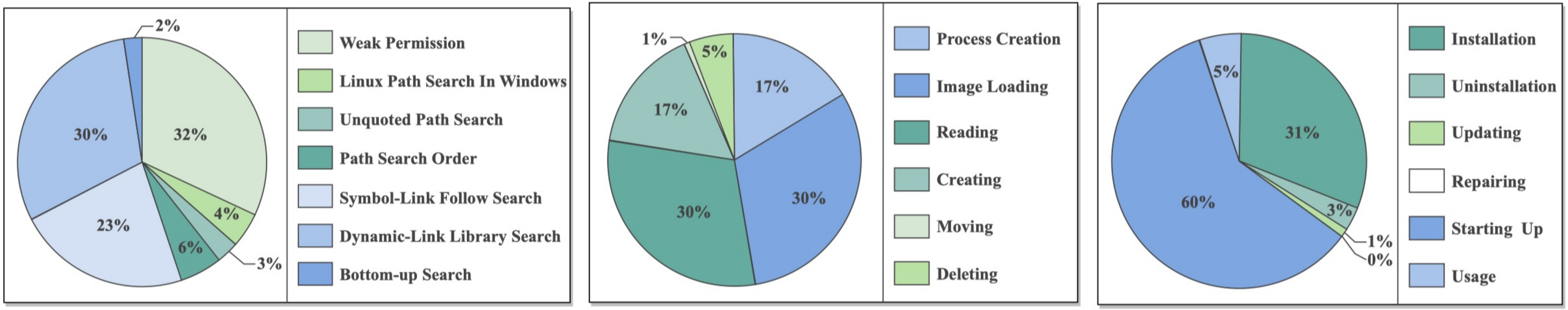
(b) File Operations

(c) Stages

Distribution of new FHVulns on different origins, file operations, and stages.

Finding 1: The bottom-up search strategy, a software-tailored search strategy that led to eight new FHVulns in fundamental software like Git and Dotnet SDK that had gone unnoticed for 18 years, has not received extensive research attention.

Analysis of New FHVulns



(a) Origins

(b) File Operations

(c) Stages

Distribution of new FHVulns on different origins, file operations, and stages.

Finding 1：The bottom-up search strategy, a software-tailored search strategy that led to eight new FHVulns in fundamental software like Git and Dotnet SDK that had gone unnoticed for 18 years, has not received extensive research attention.

Finding 2：Reading operations result in more FHVulns than we studied (30.4% vs 7.1%)，and they are more dangerous than we think.

Conclusions

- We, for the first time, provided **a clear definition of FHVuln’ s threat model**. Using this threat model, we conducted the **first empirical study** on FHVulns, revealing the origins and triggering mechanisms of FHVulns.

Conclusions

- We, for the first time, provided **a clear definition of FHVuln ‘s threat model**. Using this threat model, we conducted the **first empirical study** on FHVulns, revealing the origins and triggering mechanisms of FHVulns.

- We developed **a dynamic analysis tool, JERRY**, to detect FHVulns and applied it to **438** popular programs and uncovered **339** zero-day FHVulns. All vulnerabilities identified by JERRY were reported to the vendors, resulting in **84** of them being confirmed or fixed, with **51** CVE IDs granted and **$83,400** in bug bounties earned.

Conclusions

- We, for the first time, provided **a clear definition of FHVuln’s threat model**. Using this threat model, we conducted the **first empirical study** on FHVulns, revealing the origins and triggering mechanisms of FHVulns.

- We developed **a dynamic analysis tool, JERRY**, to detect FHVulns and applied it to **438** popular programs and uncovered **339** zero-day FHVulns. All vulnerabilities identified by JERRY were reported to the vendors, resulting in **84** of them being confirmed or fixed, with **51** CVE IDs granted and **$83,400** in bug bounties earned.

- We conducted **an in-depth analysis of the newly discovered FHVulns** and made new findings that were not observable from existing FHVulns.

Thanks for listening!
Q & A

Contact: yuchendong@iie.ac.cn